RULE BASED DECISION TREE CHATBOT WITH AI

A PROJECT REPORT

Submitted by

B. SHARAN
(Reg. No: 917721S029)

of

5 Year Integrated M.Sc. (Data Science)

in

DEPARTMENT OF APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCE

THIAGARAJAR COLLEGE OF ENGINEERING

(A Govt. Aided Autonomous Institutionaffiliated to Anna University)

MADURAI – 625 015

June 2024

THIAGARAJAR COLLEGE OF ENGINEERING, MADURAI

DEPARTMENT OF APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCE

BONAFIDE CERTIFICATE

Certified that this project report "Rule Based Decision Tree Chatbot With AI" is the bonafide work of B. SHARAN (917721S029), Seventh Semester student of 5 Year Integrated MSc (Data Science) Degree Programme, who carried out the project under my supervision from 20.06.2024 to 25.10.2024 during the academic year 2024-2025.

Head of the Department
Dr. S. Parthasarathy
Professor & Head
Department of Applied Mathematics
and Computational Science

Dr. V. Punitha
Project Guide (Internal)
Assistant Professor in Data Science
Department of Applied Mathematics
and Computational Science

Submitted for the viva – voce Examination held at Thiagarajar college of Engineering, Madurai on (05/11/2024)

___________________ ___________________

Internal Examiner External Examiner

ACKNOWLEDGEMENT

"My endeavor stands incomplete without dedicating my gratitude to a few people who have contributed a lot towards the successful completion of my project work."

I sincerely thank Almighty, for making my life to be more interesting, challenging and happier during project tenure.

I am pleased to convey my gratitude to **Dr. L. Ashok Kumar,** Principal, Thiagarajar College of Engineering, Madurai, for providing me this opportunity to do my project at ChainSys Software Exports Pvt Ltd(SEZ), Madurai.

I wish to express my sincere thanks to **Dr. S. Parthasarathy,** Head of the Department of Applied Mathematics and Computational Science, Thiagarajar College of Engineering, Madurai for his support and ardent guidance.

I am extremely thankful at the most to my internal guide **Dr. V. Punitha**, Assistant Professor, Dept. of Applied Mathematics and Computational Science, Thiagarajar College of Engineering, for her continual support and enduring guidance throughout my project tenure. My earnest thanks to all the staff members, Department of Applied Mathematics and Computational Science, for their constant care and support.

I express my faithful thanks to External Guide **Ms. Priyadharshini Nagarajan,** ChainSys Software Exports Pvt Ltd(SEZ), her guidance and expert advice rendered by her in modest attempt at preparing this project.

I wish to express my deep felt gratitude to my beloved parents for their constant support and contribution to this project work. Also, I would like to thank all my teachers, friends and well-wishers who have helped me for doing this project and throughout the 5 Year Integrated M.Sc (Data Science) course.

Sharan B

ABSTRACT

The "**Rule Based Decision Tree Chatbot with AI**" is a web-based application designed to enhance employee assistance by automating common tasks such as leave application, service request handling (including Bug, Change Request, and Support Request), and Desktop Support. This system integrates a decision tree chatbot that guides employees through various tasks, utilizing Gemini AI to handle incorrect user inputs. When an input error occurs, Gemini AI corrects the sentence, maps relevant keywords, and fetches the appropriate decision tree node stored in CouchDB, ensuring accurate responses and a smooth user experience. The chatbot simplifies processes, reducing administrative effort while improving service delivery.

Developed using Angular (HTML, CSS, TypeScript) for the frontend and CouchDB as the backend database, the system consists of two key components:

Employee Dashboard: Employees can view their personal information, leave details, and leave reports. Additionally, they can access document IDs related to service requests and desktop support. The chatbot assists employees in applying for leave and submitting requests with ease.

Admin Dashboard: Administrators have access to detailed leave summaries across the organization, with the ability to view reports for individual employees, or filter data based on day, week, month, or specific dates. This dashboard enhances transparency and helps with real-time employee management.

This system integrates a decision tree chatbot that guides employees through various tasks, utilizing Gemini AI to handle incorrect user inputs. When an input error occurs, Gemini AI corrects the sentence, maps relevant keywords, and fetches the appropriate decision tree node stored in CouchDB, ensuring accurate responses and a smooth user experience.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

S. NO.	ABBREVIATION / ACRONYM	DESCRIPTION
1	SEZ	Special Economic Zone
2	AI	Artificial Intelligence
3	HTML	HyperText Markup Language
4	CSS	Cascading Style Sheet
5	NoSQL	Non-Structured Query Language
6	SAP	Systems Applications and Products
7	PR	Performance Requirements
8	SEO	Search Engine Optimization
9	DFD	Data Flow Diagram
10	UI	User Interface
11	API	Application Programming Interface

1.INTRODUCTION

1.1. Organization Profile

ChainSys Software Exports Pvt Ltd (SEZ) is a trusted and well-established global provider of data management solutions, renowned for its long-term partnerships with technology giants such as Oracle, SAP, Microsoft, Salesforce, and others. Founded and led by CEO **Sundu Rathinam**, ChainSys focuses on delivering personalized and scalable data solutions that cater to the unique needs of each customer.

With over **26 years of industry experience**, ChainSys has developed a reputation for providing best-in-class products such as **dataZap**, **dataZen**, and **dataZense**, all designed to help businesses migrate, clean, and visualize data efficiently. The company prides itself on its in-house development of over **9000+ data accelerators**, supported by a team of **over 1100 engineers**. These efforts ensure smarter, faster, and simpler data management for their global clientele.

ChainSys follows a customer-centric approach, ensuring each engagement is personalized with customized solutions while offering ready-made tools to accelerate common problems. From **data migration and integration** to advanced **data governance and analytics**, ChainSys ensures that enterprises achieve their data transformation goals with agility and efficiency.

The company's process-driven approach begins with understanding each customer’s environment and challenges, followed by developing a tailored plan. A dedicated team ensures 24/7 support, while ChainSys continues to evolve its products to meet the changing needs of the data landscape.

With its strong partnerships and relentless focus on customer success, ChainSys remains a trusted partner to top enterprises, helping them unlock the full potential of their data.

1.2. Existing System

The existing system lacks an intelligent, interactive platform to assist employees in applying for leave, raising service requests, and managing desktop support. Current systems rely on manual processes, email communication, and inefficient tracking methods, leading to several issues:

- Lack of instant and interactive employee assistance
- Manual handling of leave requests and service issues
- Difficulty in tracking employee service requests and leave balances
- Time-consuming reporting and data retrieval processes
- Absence of AI-driven error handling in employee interactions

These limitations hinder efficiency, reduce productivity, and create unnecessary delays in processing employee requests. To overcome these challenges, an intelligent rule-based decision tree chatbot with AI is introduced.

1.3 Proposed System

The proposed system is designed to address the limitations of the existing system by implementing an interactive decision tree chatbot powered by Gemini AI. The system automates key tasks such as leave application, service requests, and desktop support, streamlining the process and improving employee experience.

1.3.1 Advantages of Proposed System

- **Planned and Organized Workflows**: The system ensures that all employee interactions are structured and organized using a decision-tree model, making it easier to navigate tasks like leave applications and service requests.
- **Error Handling through AI**: Gemini AI assists by correcting user input errors and mapping them to the correct actions, ensuring smoother and more accurate user experiences.

- **Automated Data Management**: Employee information, leave balances, and service requests are stored efficiently in CouchDB, ensuring seamless retrieval and consistent data storage.
- **Instant Retrieval of Information**: The system allows for immediate access to employee data, leave balances, and request statuses, enabling quick decision-making and reporting.
- **No Redundancy**: Information is stored consistently without duplication, optimizing data storage and ensuring data integrity.

2. SYSTEM ANALYSIS

2.1 Feasibility Study

A feasibility study assesses the practicality of the proposed system by examining its effectiveness, potential benefits, and impact on the organization. The aim of this study is to evaluate whether the proposed Rule Based Decision Tree Chatbot with AI can be successfully developed and integrated into the existing environment. The feasibility study will consider various factors such as operational, technical, and economic aspects to determine if the system is viable and beneficial to the organization.

The feasibility study is crucial because it helps identify any potential challenges that may arise during the development and deployment stages, ensuring that resources—both human and financial—are utilized effectively. In addition, it ensures that the system will meet the organization's needs, improve employee interactions, and enhance overall operational efficiency. By conducting this study, the organization can make informed decisions about the project's direction and feasibility, while also outlining steps for addressing potential risks.

2.1.1 Operational Feasibility

Operational feasibility evaluates whether the system can be successfully implemented and will function as intended within the organization. This project will provide employees with a streamlined experience for applying for leave, raising service requests, and handling desktop support through an intelligent decision-tree chatbot. By using Gemini AI for error correction and decision-making, the system reduces manual workloads and improves productivity. The system is intuitive, ensuring employees can easily adapt to its features, thus minimizing resistance and boosting acceptance.

Operationally, the system will provide instant access to important data, such as leave balances, service request statuses, and support histories, allowing employees and administrators to retrieve information quickly and make informed decisions in real-time.

2.1.2 Technical Feasibility

Technical feasibility assesses the ability of the current infrastructure to support the proposed system. The chatbot will be developed using Angular (HTML, CSS, TypeScript) for the frontend and CouchDB as the database. The AI component, powered by Gemini AI, enhances error handling and decision-making. The existing infrastructure, including any additional required resources, will be evaluated to ensure smooth performance without overloading the system.

The system is built using modern web technologies and leverages open-source tools, which ensures scalability and flexibility. The technical stack is capable of handling real-time user interactions, data retrieval, and processing, ensuring the system operates efficiently even under increased workloads.

2.1.3 Economic Feasibility

Economic feasibility examines the cost-effectiveness of the proposed system by conducting a cost-benefit analysis. The Rule Based Decision Tree Chatbot with AI relies on open-source software, including Angular, CouchDB, and Gemini AI, reducing software licensing costs. The primary costs involved will be in development, deployment, and ongoing support.

The chatbot will reduce time spent on manual tasks, improving operational efficiency and lowering administrative costs over time. The benefits of automating leave applications, service requests, and desktop support will lead to faster processing, reduced human errors, and a more productive workforce, outweighing the initial implementation and operating costs.

2.2. Use Case Diagram

2.2.1 Working of Chatbot

The figure titled "2.2.1.1 Working of Chatbot" illustrates the basic flow of a chatbot designed to assist with leave applications and service requests. Here's a breakdown of how it works:

1. **User Input:** The user interacts with the chatbot by entering their query.
2. **Process User Input:** The chatbot processes the user's input and attempts to map key words to determine the nature of the request.
3. **Keyword Mapping:**
   - If the chatbot **fails to map key words**, it responds with a message: "Sorry, I can only assist with leave applications and service requests."
   - If the chatbot **maps key words** successfully, it proceeds to the relevant service node.
4. **Service Nodes:**
   - **Leave Application Service:**
     - The chatbot welcomes the user to the leave application service.
     - The user is given options to either apply for leave or check their leave balance.
   - **Service Request:**
     - The chatbot presents the user with three options:
       - Report a bug
       - Submit a change request
       - Submit a support request.
   - **Desktop Support:**
     - The chatbot welcomes the user to the desktop support service.
5. **Database Interaction:** Once the user's request is identified, the chatbot stores the relevant details in the CouchDB database and updates the necessary records.

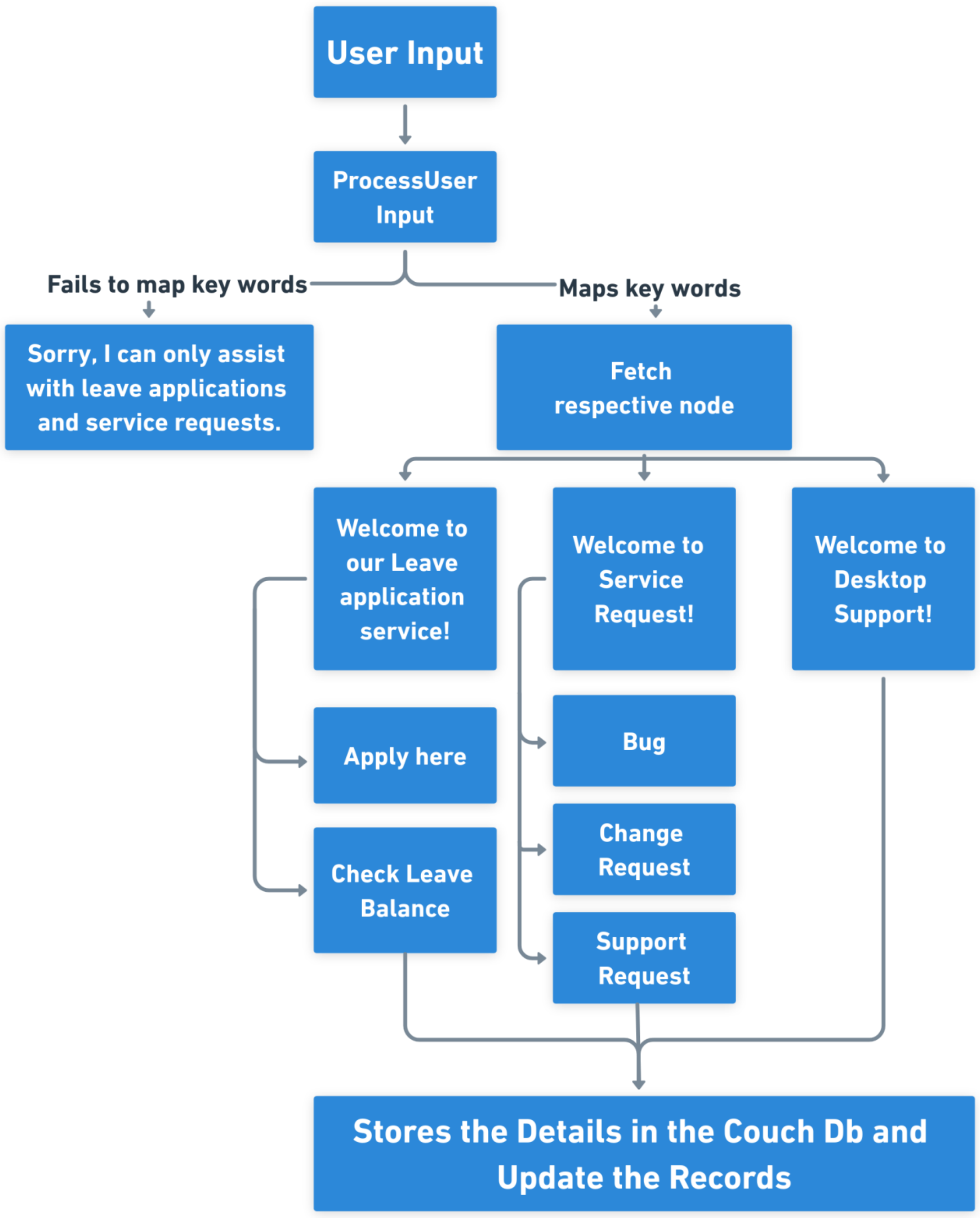

User Input
ProcessUser Input
Fails to map key words
Maps key words
Sorry, I can only assist with leave applications and service requests.
Fetch respective node
Welcome to our Leave application service!
Welcome to Service Request!
Welcome to Desktop Support!
Apply here
Check Leave Balance
Bug
Change Request
Support Request
Stores the Details in the Couch Db and Update the Records

Fig 2.2.1.1 Working of ChatBot

3. SYSTEM REQUIREMENT SPECIFICATION

3.1. Scope

The scope of this project is to develop an interactive Rule Based Decision Tree Chatbot with AI that assists employees with various organizational tasks, such as applying for leave, raising service requests (Bug, Change Request, Support Request), and managing desktop support. The chatbot will integrate Gemini AI for intelligent error correction, ensuring smooth and accurate interactions with users. The chatbot's integration with CouchDB as the database ensures robust data management, while Angular (HTML, CSS, TypeScript) enables a seamless, responsive frontend for both the employee and admin interfaces. The scope of the project is limited to enhancing employee support within the organization and ensuring accurate, real-time data tracking and reporting

3.2. Functional Requirements

The **Configuration Module** includes:

- **Configure Chatbot with CouchDB**: This function allows the configuration of the chatbot to connect with CouchDB, where it retrieves and stores user inputs, decision tree nodes, and error corrections. It ensures seamless interaction and data flow between the chatbot and the database.
- **Define Decision Tree Structure**: Administrators can create and manage the decision tree structure within the system, specifying rules, possible user inputs, and corresponding responses. This ensures that the chatbot can provide accurate guidance based on user queries.
- **Error Handling Configuration**: The system can define error correction rules that utilize Gemini AI for rephrasing user inputs when incorrect or unclear responses are detected. This feature enhances the chatbot's capability to understand user needs accurately.

The **Employee Dashboard Module** includes:

- **View Personal and Leave Details**: Employees can view their personal information, current leave balances, and detailed leave reports directly from the dashboard.
- **Raise Service Requests**: Employees can initiate service requests for Bugs, Change Requests, or Support Requests through an intuitive interface, ensuring proper documentation and tracking.
- **Raise Desktop Support**: Employees can initiate Desktop Support Requests through an intuitive interface, ensuring proper documentation and tracking

The **Admin Dashboard Module** includes:

- **View Leave Summary**: Administrators can access comprehensive leave summaries for all employees, with the ability to view detailed reports based on various criteria (date-wise, month-wise, week-wise, and daily).

The **Chatbot Interaction Module** includes:

- **Real-time Chat Interaction**: The chatbot provides instant responses to employee queries and assists in navigating through the leave application, service request and desktop support process.
- **Keyword Mapping and Decision Tree Navigation**: When users provide input, the chatbot maps the keywords to the corresponding nodes in the decision tree stored in CouchDB, ensuring accurate responses and guidance.

3.3. Performance Requirements

Performance Requirements (PR) are crucial for the design and development of the system. There are three primary classes of performance requirements: **response time** (the speed at which the system processes individual requests), **throughput** (the number of requests the system can handle), and **concurrency** (the number of simultaneous users or threads). The performance requirements for this project are outlined below:

3.3.1 Reusability

The system is designed with modularity in mind, utilizing reusable components and classes. This approach allows developers to efficiently integrate new functionalities with minimal modifications to the existing codebase, thereby reducing implementation time.

3.3.2 Flexibility

The proposed system is developed using Angular and CouchDB, allowing for easy modifications to accommodate operational changes. This flexibility ensures that the system can adapt to evolving user needs and business requirements without extensive rework.

3.3.3 Schedule operation Reliability

The system is designed for high availability and reliability, ensuring a failure-free operation. It will be accessible to users at all times, minimizing downtime and enhancing user experience.

3.3.4 Performance

The proposed system will provide fast response times to user inputs, ensuring minimal delay for each operation performed. This responsiveness is vital for maintaining user engagement and satisfaction.

3.3.5 Functionality

The system will include all the essential functionalities outlined in the requirements, ensuring interoperability and accuracy. It will effectively manage user interactions, decision tree navigation, and data retrieval.

3.3.6 Speed

The system is optimized for quick result generation, minimizing wait times for users. By eliminating unnecessary steps and avoiding the recording of irrelevant information, the system streamlines processes to enhance performance. The complexity of various processes is managed by breaking them into distinct modules, enabling efficient processing. With a centralized database hosted on a server, rapid data processing and retrieval are facilitated, contributing to improved overall performance and speed.

3.4 Software Specification

SOFTWARE	VERSION
Angular	15.2.9
Node	18.17.0
npm	9.6.7

Table 3.4.1 Software Requirements

3.5 Hardware Specification

Unit	Configuration (recommended)
Processor	Intel® Core™ i5-8265U CPU @ 1.60GHz 1.80 GHz
Memory	At least 512 MB
Hard disk	At least 10 GB
System Type	64-bit operating system, x64-based processor
Ethernet Card	Basic Model

Table 3.5.1 Hardware Configuration

3.6 Technologies Used

Technologies: HTML, CSS, TypeScript, NoSQL

3.6.1 HTML

HTML, or HyperText Markup Language, serves as the primary markup language for crafting web pages and displaying content in web browsers. It is structured with HTML elements formed by tags enclosed in angle brackets, creating a hierarchical structure within web content. HTML is foundational to all websites, allowing for the embedding of images and objects, as well as the creation of interactive forms. It denotes structural semantics for various text elements, including headings, paragraphs, lists, links, and quotes.

Moreover, HTML can integrate scripts written in languages like JavaScript, which can modify the behavior of web pages. Web browsers can also utilize Cascading Style Sheets (CSS) to define the visual presentation and layout of the content. By providing a clear structure, HTML plays a crucial role in SEO (Search Engine Optimization), making it easier for search engines to index content effectively. Its evolving standards, such as HTML5, also support multimedia elements like video and audio, enhancing the interactivity and richness of web applications.

3.6.2 CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in HTML. CSS enhances the aesthetic appeal of web pages by allowing developers to separate content from design, facilitating a more organized approach to web development. It enables the application of styles such as fonts, colors, spacing, and layouts, ensuring consistency across multiple pages. CSS also supports responsive design, making it easier to adapt the website's appearance on various devices and screen sizes, thereby improving the user experience. The ability to use media queries also allows developers to create tailored experiences for different devices, ensuring that users have a seamless experience whether they are on a desktop, tablet, or smartphone.

3.6.3 TypeScript

TypeScript is a superset of JavaScript that adds static typing to the language, which helps in catching errors during development and improving code quality. It allows developers to write robust, maintainable code while still benefiting from JavaScript's flexibility. TypeScript's strong typing system helps in defining interfaces and ensures that the right data types are used, which is particularly beneficial in larger projects where collaboration is key. With TypeScript, developers can utilize advanced features such as classes and modules, enhancing the overall structure and organization of the codebase.

This leads to improved readability and easier debugging, as type errors can be identified at compile time rather than runtime. Furthermore, TypeScript's integration with modern development tools and frameworks like Angular significantly boosts productivity, allowing for more efficient development cycles and better collaboration among teams.

3.6.4 NoSQL

NoSQL databases, such as CouchDB used in this project, provide a flexible and scalable alternative to traditional relational databases. Designed to handle unstructured and semi-structured data, NoSQL databases allow for easy storage and retrieval of large volumes of diverse data types. They excel in scenarios where high availability and horizontal scalability are critical, such as in web applications. The schema-less nature of NoSQL databases allows developers to quickly adapt to changing data models, making it easier to manage evolving application requirements. In this project, CouchDB facilitates efficient data management for the chatbot, enabling real-time updates and interaction. Additionally, NoSQL databases typically support distributed architectures, allowing for better load balancing and fault tolerance, which enhances the overall resilience of the application. The use of document-oriented storage in CouchDB aligns well with the needs of modern web applications, where data structures can vary significantly, enabling agile development practices.

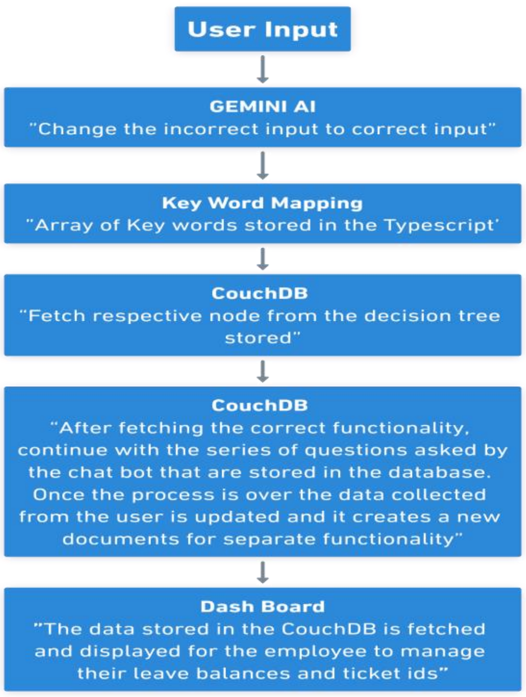
4. SYSTEM DESIGN

System design is a critical process that serves as the backbone of developing any software product. The design phase translates the insights and results gathered during the analysis phase into a blueprint for the system. This phase is essential for determining how the various components of the software will interact, how data will flow, and how user interactions will be handled. A well-defined system design not only guides developers in creating the software but also ensures that the end product aligns with user needs and project objectives.

The output of the design phase is often a prototype that closely resembles the final product, allowing stakeholders to visualize the system's functionality and interface. This iterative approach helps in identifying potential issues early in the development process and facilitates adjustments based on user feedback.

4.1 System Architecture Diagram

The Fig 4.1 system architecture diagram provides a high-level overview of the system components, their interactions, and how data flows through the application. It outlines the key elements of the architecture, including front-end interfaces, back-end services, databases, and external integrations.



User Input
GEMINI AI
"Change the incorrect input to correct input"
Key Word Mapping
"Array of Key words stored in the Typescript'
CouchDB
"Fetch respective node from the decision tree stored"
CouchDB
"After fetching the correct functionality, continue with the series of questions asked by the chat bot that are stored in the database. Once the process is over the data collected from the user is updated and it creates a new documents for separate functionality"
Dash Board
"The data stored in the CouchDB is fetched and displayed for the employee to manage their leave balances and ticket ids"

Fig 4.1 Architecture Diagram

4.2 DATA FLOW DIAGRAM

Dataflow is the movement of data in a system from a point of origin to a specified destination indicated by line or arrow. Dataflow diagram is the graphical representation of the data movements, processes and files (data stores) used in support of information systems.

The data flow diagram (DFD) is one of the most important tools used by system analysts. Data flow diagrams are made up of a number symbols, which represent system components. Most data flow modeling methods use four kinds of symbols. These symbols are used to represent four kinds of system components. Processes, data stores, data flows and external entities.

The Figure 4.2 User Input section explains how the chatbot collects and processes information from users. As the chatbot interacts with the user by asking a series of questions, the responses are stored as data. This data is then organized and updated within the CouchDB database. For each specific functionality, such as leave applications or service requests, the system creates new documents in CouchDB to store the information, ensuring that each transaction is independently documented and can be easily retrieved later.

In the Dash Board section, the description highlights the process of retrieving data stored in CouchDB. The dashboard fetches this data and presents it to employees, allowing them to manage their information. For instance, employees can view and manage their leave balances or check the status of their ticket IDs related to service requests or desktop support.

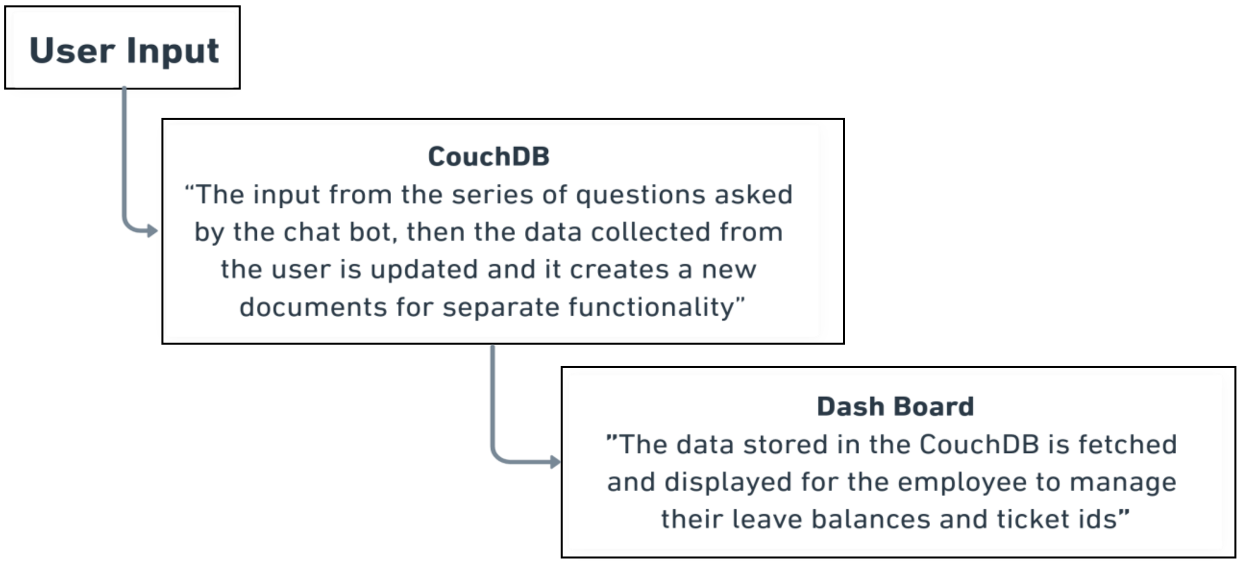

User Input
CouchDB
“The input from the series of questions asked by the chat bot, then the data collected from the user is updated and it creates a new documents for separate functionality”
Dash Board
”The data stored in the CouchDB is fetched and displayed for the employee to manage their leave balances and ticket ids”

Fig 4.2 DFD

4.3 ER DIAGRAM

The Figure 4.3 illustrates the relationships between an Employee and various functionalities such as Leave, Leave Details, Desktop Support and Service Requests in a database system:

1. Employee: Stores key employee details like `employeeId`, `employeeName`, contact information, and leave eligibility.

2. Leave: Tracks the employee's leave balances (casual, medical, etc.) and records leaves taken.

3. Leave Details: Logs individual leave requests with details such as leave type, reason, start/end dates, and total leave days.

4. Desktop Support: Captures employee desktop support requests, including module, title, contact number, and issue details.

5. Service Request: Manages service tickets for technical issues, logging information like severity, priority, environment, and issue resolution steps.

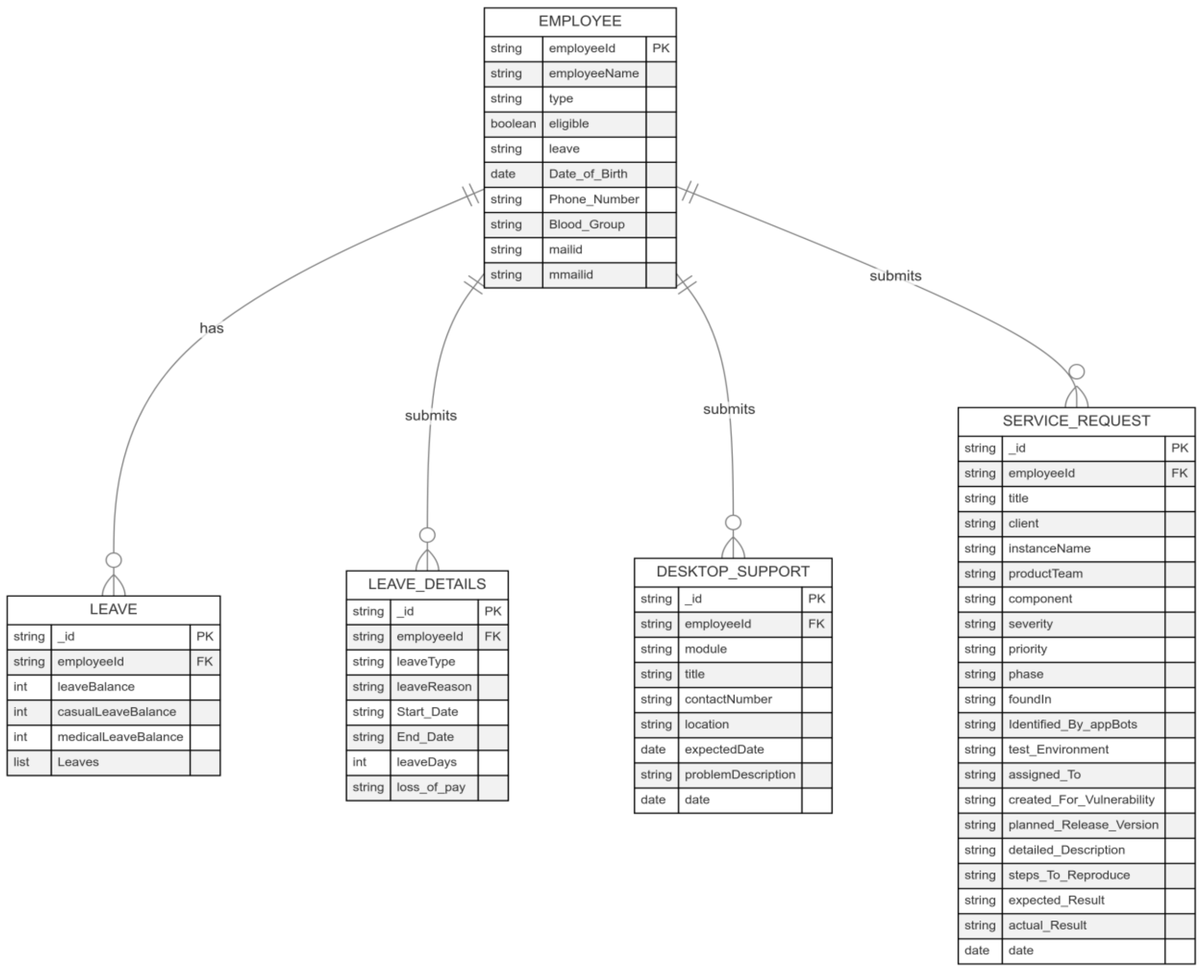
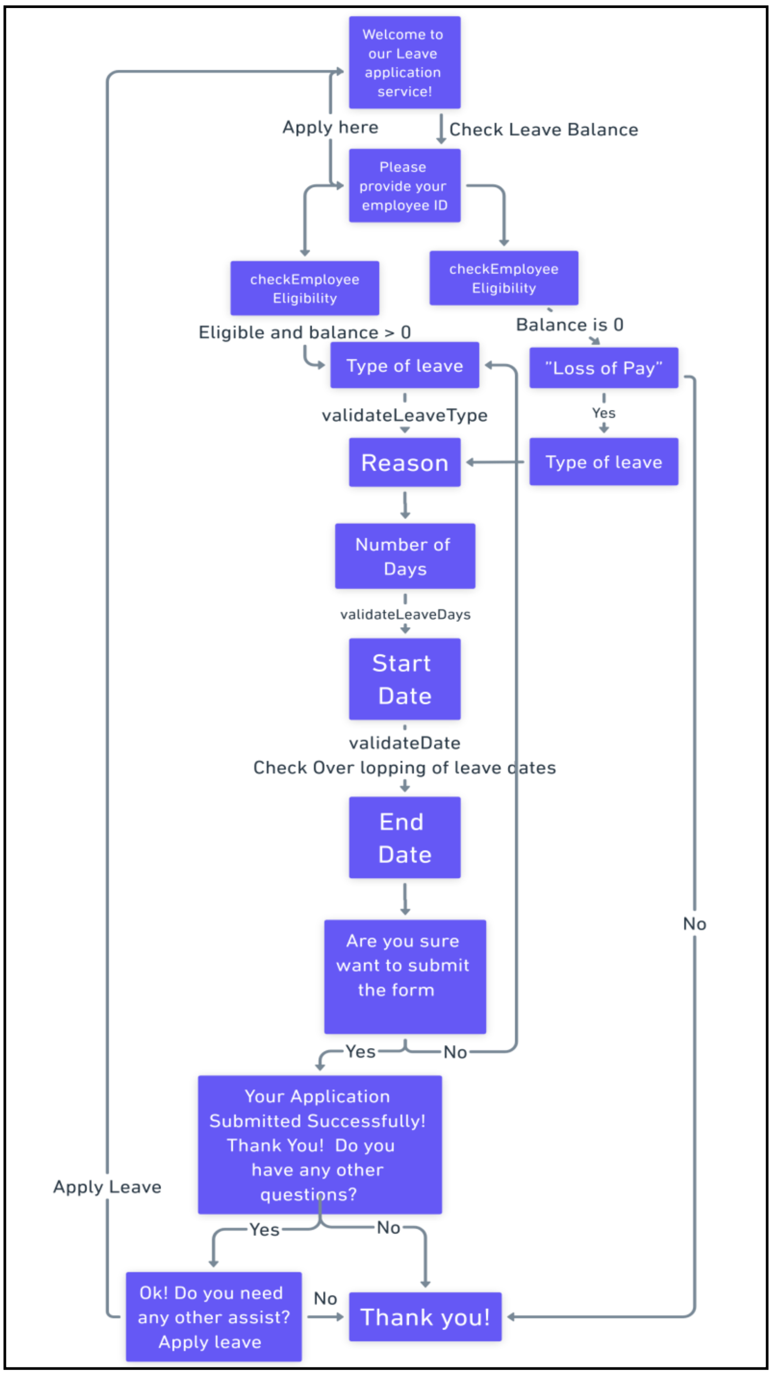


Fig 4.3 Entity Relationship Diagram

4.4 PROCESS FLOW DIAGRAM

The Figure 4.4.1, flow guides the employee through a leave application process, starting with checking leave balance and eligibility. If eligible, it collects details like the type of leave, reason, number of days, and the start and end dates. After validation, the leave is submitted. If the leave balance is zero, it marks the leave as "Loss of Pay." The flow ends with a confirmation and offers further assistance.



Welcome to our Leave application service!
Apply here
Check Leave Balance
Please provide your employee ID
checkEmployee Eligibility
checkEmployee Eligibility
Eligible and balance > 0
Balance is 0
Type of leave
"Loss of Pay"
validateLeaveType
Yes
Reason
Type of leave
Number of Days
validateLeaveDays
Start Date
validateDate
Check Over lopping of leave dates
End Date
Are you sure want to submit the form
Yes
No
No
Your Application Submitted Successfully! Thank You! Do you have any other questions?
Apply Leave
Yes
No
Ok! Do you need any other assist? Apply leave
No
Thank you!

Fig 4.4.1 Process Flow Diagram for Leave Application

The Figure 4.2.2 represents a process for raising a service request – Bug. It starts with the employee entering their ID and details like title, client, instance name, and component. It then captures additional information such as priority, severity, environment details, and description of the issue. The request is assigned, and the user can upload a screenshot for further clarification.

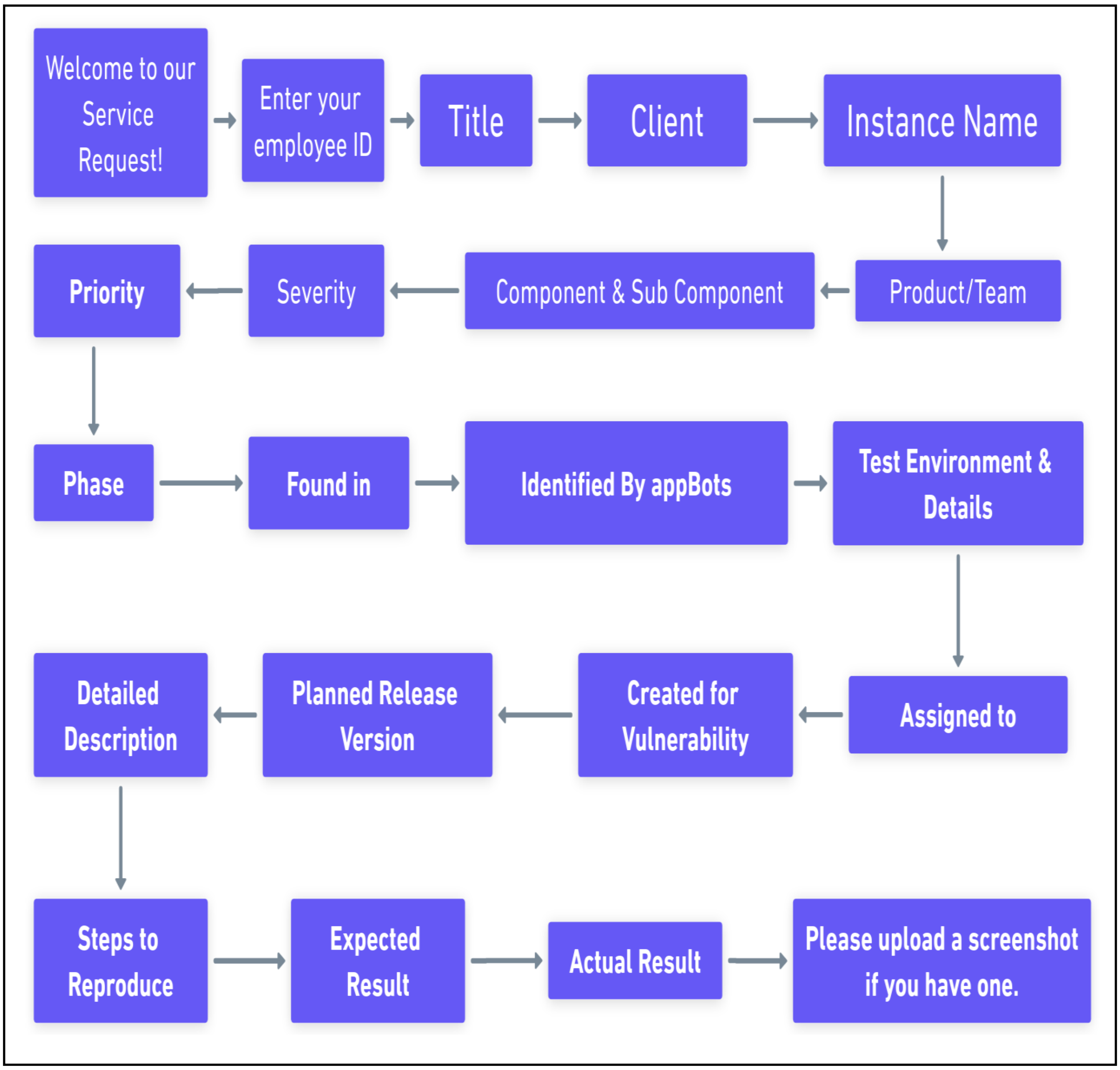


Fig 4.4.2 Process Flow Diagram for SR - Bug

The Figure 4.4.3 represents a streamlined service request – Change Request flow. It starts with collecting the employee's ID, title, and client details, followed by priority, severity, and product information. It captures relevant environment details, the planned release version, and allows the user to upload screenshots.

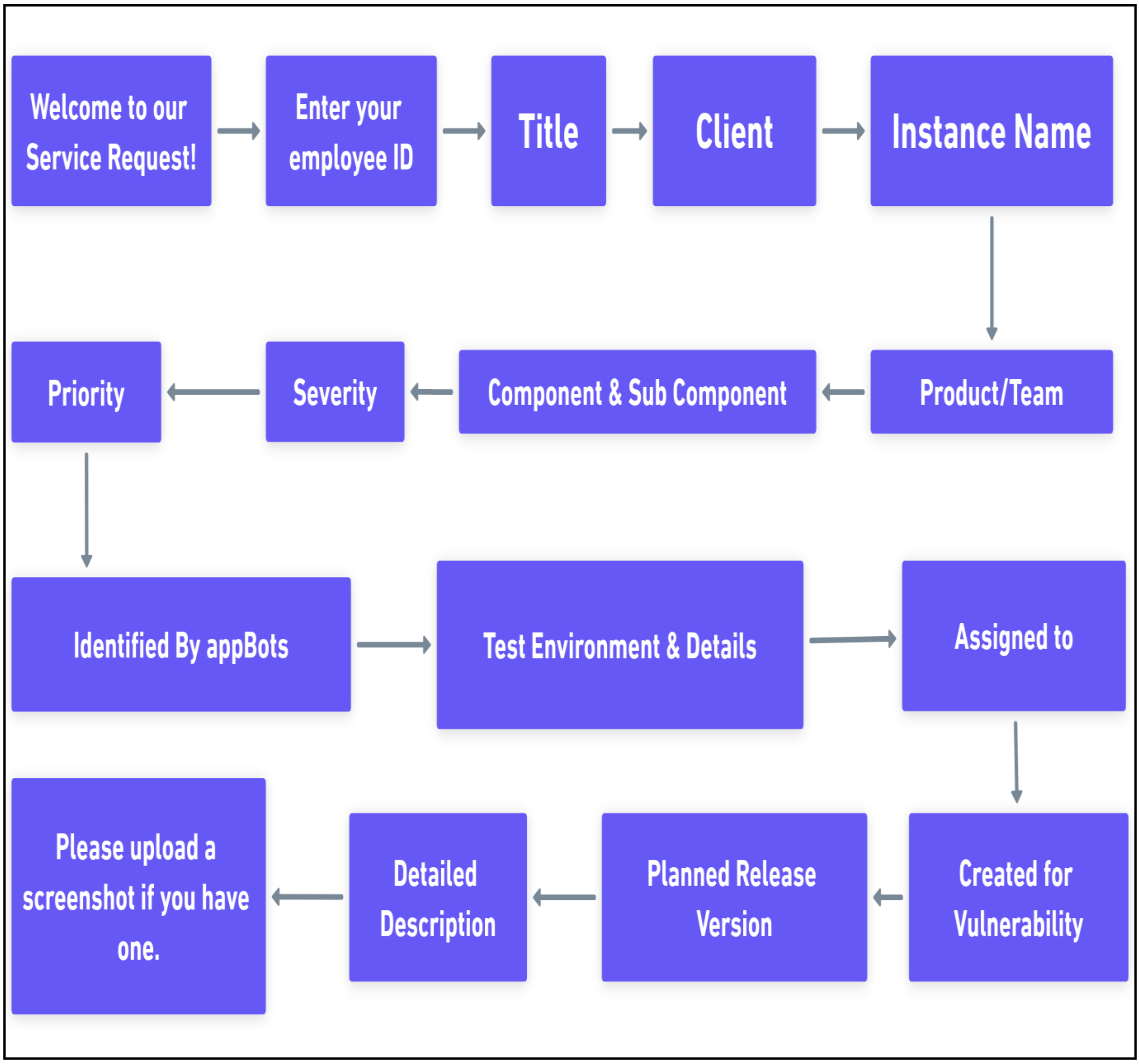

Welcome to our Service Request!
Enter your employee ID
Title
Client
Instance Name
Product/Team
Component & Sub Component
Severity
Priority
Identified By appBots
Test Environment & Details
Assigned to
Created for Vulnerability
Planned Release Version
Detailed Description
Please upload a screenshot if you have one.

Fig 4.4.3 Process Flow Diagram for SR – Change Request

The Figure 4.4.4 represents a streamlined service request – Support Request flow. It starts with collecting the employee's ID, title, and client details, followed by priority, and product information. It captures relevant environment details, the planned release version, and allows the user to upload screenshots.

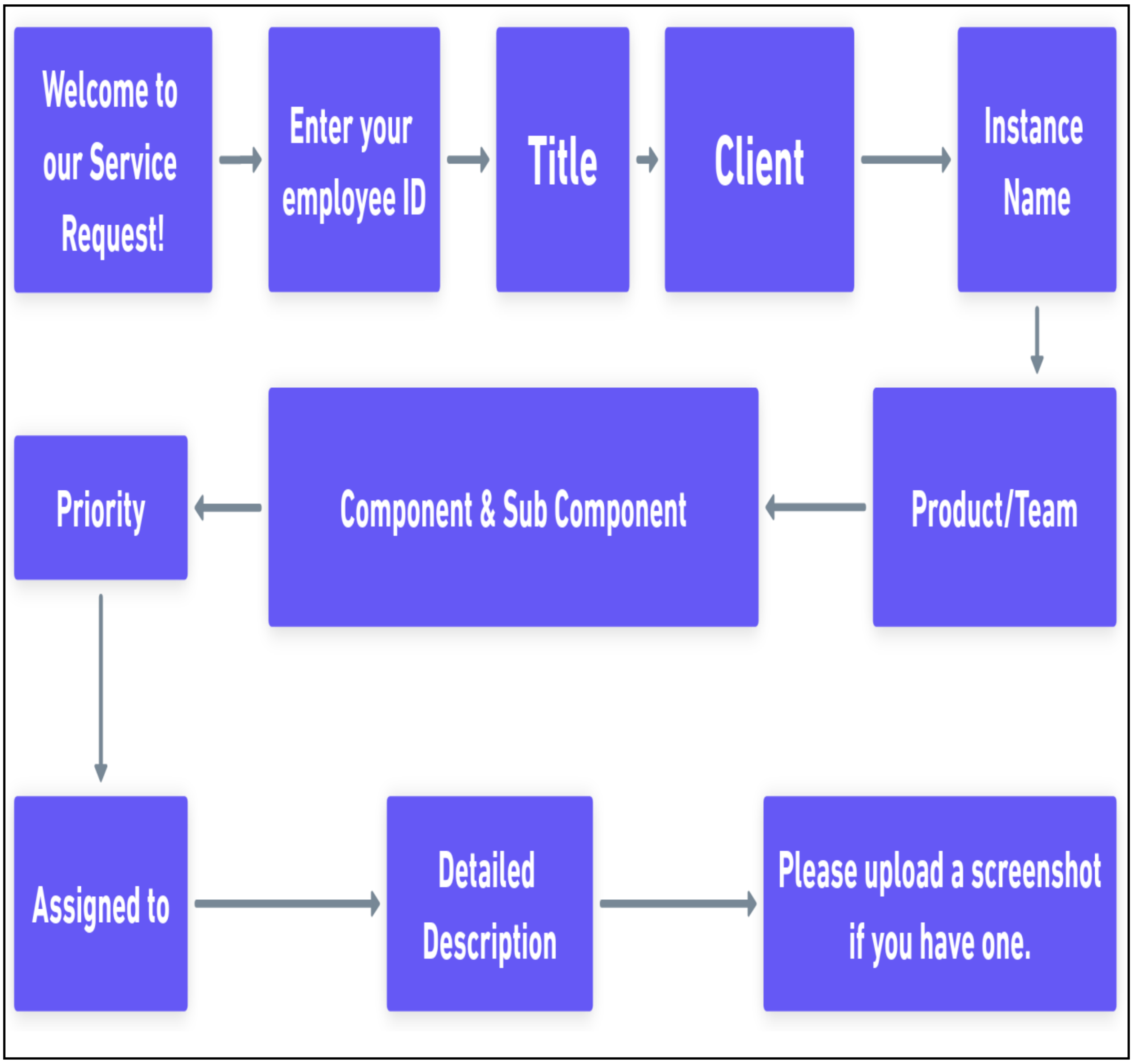

Welcome to our Service Request!
Enter your employee ID
Title
Client
Instance Name
Product/Team
Component & Sub Component
Priority
Assigned to
Detailed Description
Please upload a screenshot if you have one.

Fig 4.4.4 Process Flow Diagram for SR – Support Request

The Figure 4.4.5 represents a streamlined Desktop Support flow. It starts with collecting the employee’s ID, title, and client details, followed by priority, severity, and product information. It captures relevant environment details, the planned release version, and allows the user to upload screenshots.

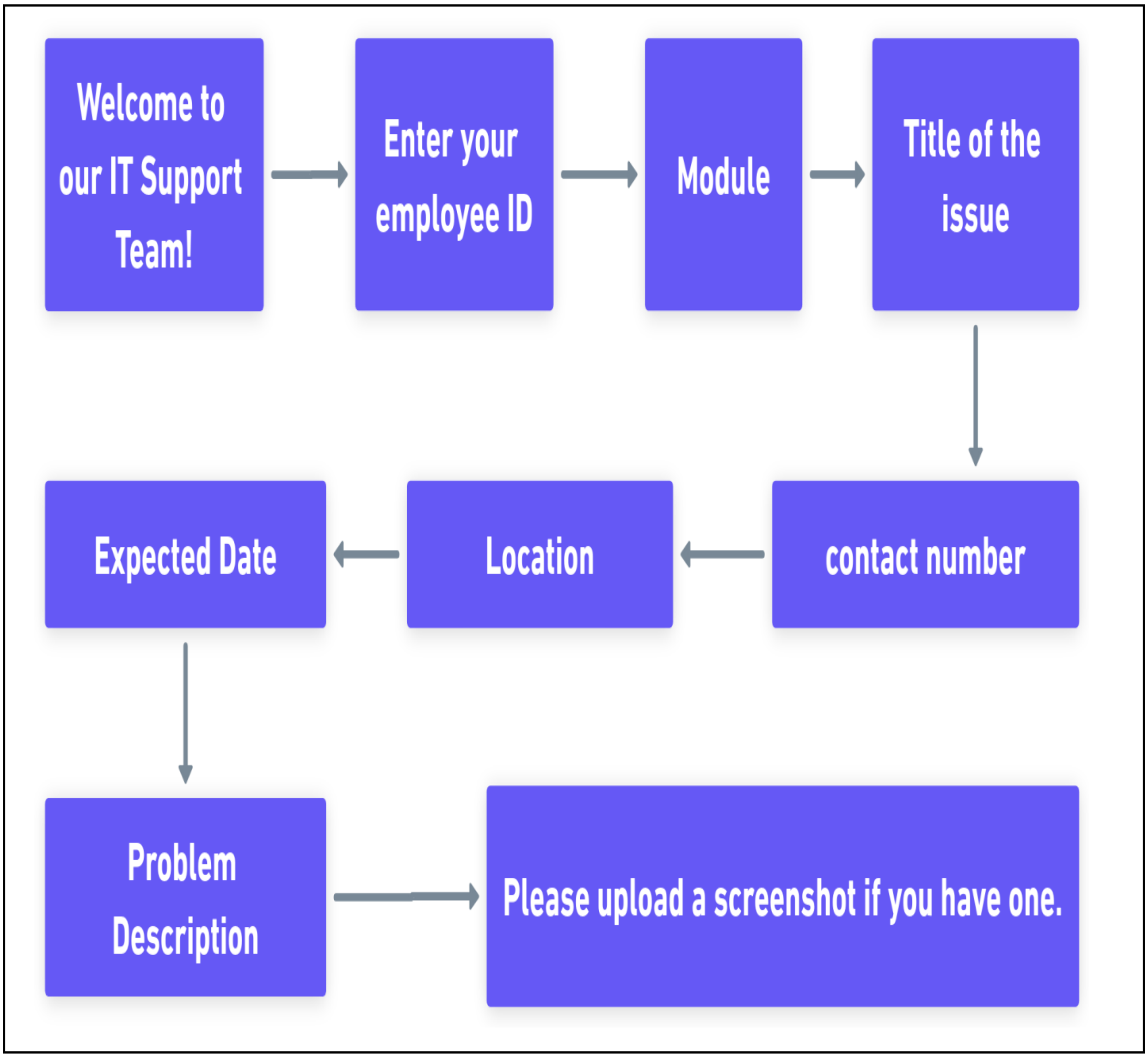


Fig 4.4.5 Process Flow Diagram for Desktop Support Request

4.5 PHYSICAL DESIGN USING UML:

4.5.1 Component Diagram:

The project features a comprehensive system with both an **Employee Dashboard** and an **Admin Dashboard**, designed to streamline leave management desktop support and service requests. The **Login Component** validates user credentials, redirecting employees to the Employee Dashboard and administrators to the Admin Dashboard. **The Employee Dashboard** allows employees to view and manage their leave details, service request documents, and desktop support documents. Additionally, the ChainBot component offers online assistance, guiding employees through the leave application, desktop support and service request processes. On the other hand, the **Admin Dashboard** enables administrators to access real-time Leave Summaries, including reports categorized as Today, Date-wise, Month-wise, and Week-wise, as well as summaries for all employees or specific employees by ID. This system enhances organizational efficiency by providing a centralized platform for managing employee records and offering responsive online assistance.

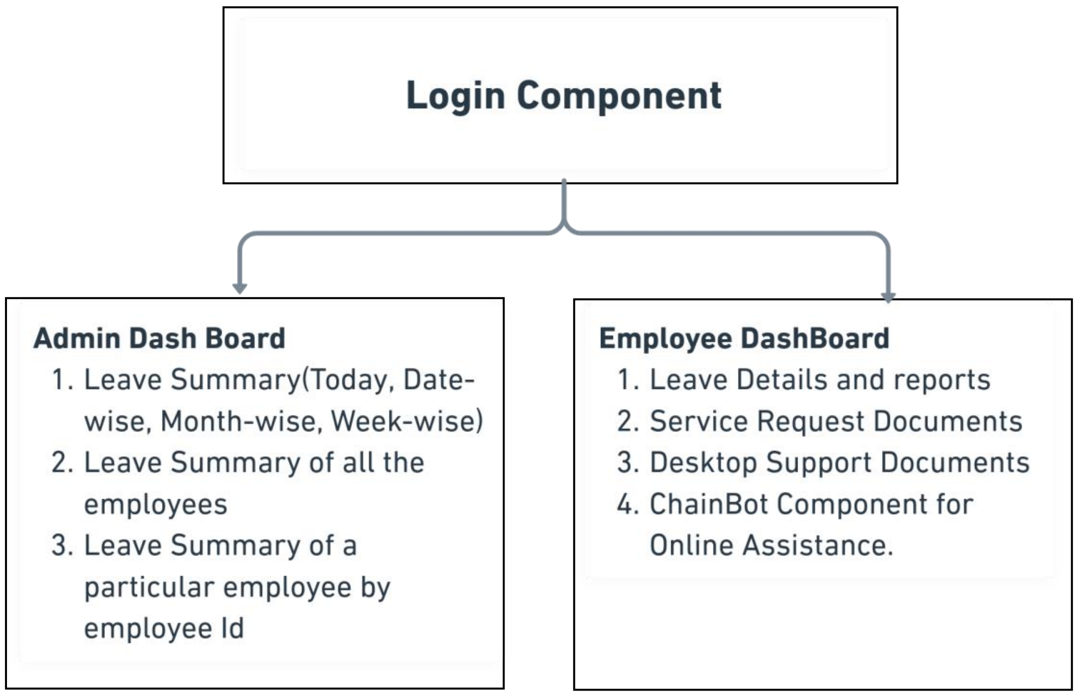

Login Component
Admin Dash Board
1. Leave Summary(Today, Date-wise, Month-wise, Week-wise)
2. Leave Summary of all the employees
3. Leave Summary of a particular employee by employee Id
Employee DashBoard
1. Leave Details and reports
2. Service Request Documents
3. Desktop Support Documents
4. ChainBot Component for Online Assistance.

Fig 4.5.1 Component Diagram

4.6 MODULE DESCRIPTION:

4.6.1 Employee Dashboard:

4.6.1.1 Home:

The central point of the dashboard, the home section allows you to return to the main dashboard view. From here, you can see all available options and navigate to different features of the home dashboard and how to use ChainBot.

4.6.1.2 My Details:

Allows users to view their personal employee information.

4.6.1.3 Leave Menu:

This button toggles a dropdown for leave-related options:

4.6.1.3.1 Leave Details: Users can track their leave balances, including Casual and Medical Leave.

4.6.1.3.2 Leave Report: Users can review detailed records of their leave usage, organized by year and including types and reasons.

4.6.1.4 Service Requests:

This option provides access to service requests, where users can view their SR IDs and data.

4.6.1.5 Desktop Support:

This option provides access to support tickets related to desktop issues.

4.6.1.6 ChainBot:

Chainbot, Online Assistant. I’m here to help you with Leave Application, Desktop Support and Service Requests.

4.6.2 Admin Dashboard:

4.6.2.1 Home:

The central point of the dashboard, the home section allows you to return to the main dashboard view. From here, you can see all available options and navigate to different features of the admin dashboard.

4.6.2.2 Leave Summary:

This section provides a high-level overview of employee leave data, segmented by time periods:

No. of Employees took leave Today: It is a title card that shows the count of employees who have taken leave today and click to view the date wise Leave Summary.

No. of Employees took leave this Week: It is a title card that shows the count of employees who have taken leave this week and click to view the week wise Leave Summary.

No. of Employees took leave this Month: It is a title card that shows the count of employees who have taken leave this Month and click to view the month wise Leave Summary.

No. of Employees took leave this Year: It is a title card that shows the count of employees who have taken leave this Year.

4.6.2.3 Leave Summary of all Employees:

Here, you can check the detailed leave records for all employees. This feature allows for tracking, enabling you to manage an employee’s leave balance.

4.6.2.4 Summary of a particular Employee:

Here, you can check the detailed leave records for any specific employee. This feature allows for individualized tracking, enabling you to manage an employee’s leave balance, review their leave history, and ensure they are adhering to company leave policies.

5. TESTING

5.1 TESTING OBJECTIVE

Software testing is a critical phase in the development of the system. It primarily focuses on validating the integration of various modules within the system. The main objectives of testing are:

- **Ensure Alignment with Objectives**: The primary aim is to identify any discrepancies between the developed system and the original requirements, ensuring that the final product meets the established goals.
- **Quality Assurance**: The quality of an information system is contingent upon its design, development, and implementation. Testing serves as a vital activity in this development phase, as it directly influences the reliability and performance of the system.
- **Error Detection**: Testing is fundamentally about discovering errors or bugs within the system. This process involves identifying defects that could impede functionality and usability.
- **User Satisfaction**: Ultimately, testing ensures that the needs and expectations of the users are met. It acts as a validation process to confirm that the system performs as intended.

Types of Testing Conducted

1. **Unit Testing**: This involves testing individual components or modules of the system to ensure they function correctly in isolation.
2. **Integration Testing**: This testing phase assesses the interactions between different modules to verify that they work together seamlessly.
3. **User Interface Testing**: This focuses on evaluating the usability and user experience of the system's interface, ensuring it is intuitive and accessible.
4. **Functional Testing**: This tests the system's functionalities against the specified requirements, ensuring all features perform as expected.
5. **System Testing**: This comprehensive testing phase evaluates the entire system's compliance with the requirements and checks for any potential issues before deployment.

5.2 UNIT TESTING

Unit testing is a software testing technique that focuses on verifying the correctness of individual components or modules of a software application. Each unit, typically a small part of the code, such as a function or method, is tested in isolation from the rest of the system.

Sample Test cases:

S. No.	**Test Condition Description**	**Test Case No.**	**Test Case/Scenario/Data Description**	**Expected Result**	**Actual Result**	**Pass/Fail**
1	Check fetching and displaying employee details	**1.1**	Validate data fetching from CouchDB.	Employee details fetched successfully.	Employee details fetched successfully.	Pass
2	Check fetching and displaying leave balances	**1.2**	Check correct display of employee leave balance.	Leave balance displayed correctly.	Leave balance displayed correctly.	Pass
3	Check employee service requests	**1.3**	Verify accurate display of service requests on the dashboard.	Service requests displayed correctly.	Service requests displayed correctly.	Pass
4	Validate CRUD operations in CouchDB	**2.1**	Test create operation for employee records.	Employee record created successfully.	Employee record created successfully.	Pass
5	Validate data retrieval in CouchDB	**2.2**	Validate read operation for fetching leave data.	Leave data fetched successfully.	Leave data fetched successfully.	Pass
6	Check update and delete operations in CouchDB	**2.3**	Check update operation for service requests.	Service request updated successfully.	Service request updated successfully.	Pass
7	Validate user login	**3.1**	Validate successful login for admin role.	Admin logged in successfully.	Admin logged in successfully.	Pass
8	Check redirection after login	**3.2**	Check redirection for employee role after login.	Redirected to employee dashboard.	Redirected to employee dashboard.	Pass

9	Test leave balance calculation	**4.1**	Test calculation with various leave usage scenarios.	Remaining leave calculated correctly.	Remaining leave calculated correctly.	Pass
10	Validate employee ID eligibility	**4.2**	Validate responses for valid and invalid employee IDs.	Correct eligibility response displayed.	Correct eligibility response displayed.	Pass
11	Validate input fields in forms	**5.1**	Test validation for employee ID and leave dates.	Error message for invalid input displayed.	Error message for invalid input displayed.	Pass
12	Check UI responsiveness	**6.1**	Test button functionality and form submission.	Button functions as expected.	Button functions as expected.	Pass
13	Validate bot response for zero leave balance	**7.1**	Test responses for leave applications with zero balance.	Bot indicates leave will be marked as loss of pay.	Bot indicates leave will be marked as loss of pay.	Pass
14	Assess system stability with multiple requests	**7.2**	Simulate concurrent leave requests.	System handles concurrent requests smoothly.	System handles concurrent requests smoothly.	Pass
15	Check event handlers for user interactions	**8.1**	Test button clicks and form submissions.	Correct functions triggered upon interaction.	Correct functions triggered upon interaction.	Pass
16	Validate functionality with mocked CouchDB responses	**9.1**	Validate component and service functionality without actual DB.	Component works as expected with mocked data.	Component works as expected with mocked data.	Pass

Table 5.1 Unit Test Cases

5.3 INTEGRATION TESTING

Integration testing is a systematic technique used to construct the program structure while simultaneously testing for errors that may occur during the integration of unit-tested modules. Once unit testing is completed, the modules are combined to form a complete system, and testing is performed to identify any discrepancies between the individual components. The goal of integration testing is to ensure that these components work cohesively and the overall system functions as expected.

In this project, the following key modules are integrated and tested:

- Configuration Module
- Representative Setup
- Call Tracking
- Desktop Functions

The testing for these modules ensures that the data flow between them is seamless and the individual functionalities of each module do not adversely affect the performance or functionality of others.

5.3.1 Key Focus Areas of Integration Testing:

- **Data Flow Between Modules:** Even though modules may pass unit tests individually, data passing between modules can cause unexpected errors. The integration testing ensures that information shared between modules is accurate and handled correctly.
- **Intermodule Interactions:** The integration process focuses on interactions between modules to ensure that one module does not negatively impact another, and that combined sub-functions produce the desired major function.
- **Global Data Structures:** During integration testing, special attention is given to global data structures, as improper handling may lead to data inconsistency across the system.
- **Internal and External Interfaces:** The emphasis is on testing the interfaces between internal modules and external interfaces, such as the communication between the frontend (Angular) and CouchDB backend, ensuring that they function correctly when integrated.

5.3.2 The following modules are integrated:

- **Configuration Module:** Ensures that the system configurations are correctly applied and shared with other modules.
- **Representative Setup:** Ensures that employee-related configurations are properly passed to other components such as leave management or service requests.
- **Call Tracking:** Integration of the call tracking system ensures that the system records and tracks leave application, service requests, and desktop support requests with data flowing seamlessly into the Desktop Functions module.
- **Desktop Functions:** This module interacts with leave application, service requests, and desktop support requests. Integration testing ensures that these interactions are correct and do not lead to discrepancies in data storage or performance.

5.3.3 The integration process focuses on uncovering issues such as:

- Errors in data transfer between modules.
- Adverse effects of one module on another.
- Proper execution of combined sub-functions.
- Validation of system performance based on the design specifications.

Thus, during integration testing, the modules are verified for proper interaction, and any errors identified are rectified to ensure smooth performance and data integrity across the entire system. The performance levels established during the design phase are also validated during this testing phase.

Sample Test cases:

S. No.	Test Condition Description	Test Case No.	Test Case/Scenario/Data Description	Expected Result	Actual Result	Pass/Fail
1	Validate data flow between Configuration Module and Representative Setup	1.1	Ensure that configuration settings are properly passed to the representative setup module.	Configuration data shared successfully.	Configuration data shared successfully.	Pass
2	Check data transfer between Representative Setup and Call Tracking	1.2	Validate those representative details are properly communicated to the call tracking module.	Representative details transferred without issues.	Representative details transferred without issues.	Pass
3	Test integration between Call Tracking and Desktop Functions	1.3	Ensure that call tracking data is properly fed into the Desktop Functions module.	Call tracking data transferred and utilized correctly.	Call tracking data transferred and utilized correctly.	Pass
4	Verify interaction between Desktop Functions and Service Requests	1.4	Ensure that service requests (bugs, change requests, etc.) are correctly updated in Desktop Functions.	Service requests integrated and displayed in Desktop Functions.	Service requests integrated and displayed.	Pass
5	Check overall system performance after module integration	1.5	Validate the system's ability to handle combined functionalities across all modules.	System performance remains consistent and meets design specifications.	System performance remains consistent.	Pass
6	Test global data structures and ensure proper integration	1.6	Verify that global data structures are correctly shared across all modules.	Data consistency maintained throughout modules.	Data consistency maintained.	Pass

7	Test inter-module interaction for errors	1.7	Simulate potential errors in one module and verify if it affects other modules.	Errors isolated to the specific module and do not affect others.	Errors isolated and handled appropriately.	Pass
8	Verify performance levels established during software design	1.8	Ensure that all integrated modules meet the performance benchmarks defined during the design phase.	All modules perform within the expected thresholds.	All modules perform as expected.	Pass

Table 5.2 Integration Test Cases

5.4 USER INTERFACE TESTING

User Interface (UI) Testing ensures that the interface of the system functions as per the requirements and provides a seamless experience for the user. UI testing focuses on validating the visual elements and interactions of the system such as buttons, forms, navigation, and responsiveness. In your project, UI testing will focus on:

- **Employee Dashboard**: Verify that the employee's personal details, leave balances, and service requests are displayed correctly and are easy to navigate.
- **Admin Dashboard**: Ensure that leave summaries, reports, and filters for various employees are presented in a user-friendly manner.
- **Form Validation**: Check that all form inputs for leave requests, service requests, and desktop support are correctly validated and provide appropriate error messages for invalid inputs.
- **Responsiveness**: Verify that the application is responsive and functions well on various screen sizes (desktop, tablet, mobile).

5.5 PERFORMANCE TESTING

Performance Testing aims to assess the system’s behavior under specific conditions, ensuring it can handle peak loads and performs efficiently under stress. In your project, performance testing will include:

- **Load Handling**: Test how the system performs when multiple users are applying for leave, submitting service requests, or viewing reports simultaneously.
- **Navigation Testing**: Evaluate the speed of navigation between different modules (Employee Dashboard, Admin Dashboard, Leave Application, Service Requests).
- **Response Time**: Measure the time it takes for the system to fetch data from CouchDB (e.g., employee details, leave balances, service requests) and display it to the user.
- **System Latency**: Test the latency when multiple service requests are raised at once or when generating reports for all employees.

5.6 SYSTEM TESTING

System Testing focuses on validating the interaction between all the integrated modules in the system and ensuring that the project as a whole meets the defined requirements. In your project, system testing will involve:

- **Module Compatibility**: Ensure that all integrated modules (Employee Dashboard, Admin Dashboard) work seamlessly together.
- **End-to-End Testing**: Test the entire flow from an employee logging in, applying for leave, submitting service requests, to an admin reviewing data.
- **Error Detection**: Identify discrepancies between the system's actual performance and its original design specifications.
- **Final User Verification**: Ensure that after all modules are integrated and tested, the system satisfies the needs of the end users (employees and admins).

System testing is the final step before deploying the system, ensuring that the overall performance and functionality meet the project’s objectives and user expectations.

6. IMPLEMENTATION

The Implementation phase is a critical step in the software development life cycle, as it focuses on converting the design and specifications into a functional and working system. This phase involves coding, testing, and installing the software, and ensuring that it operates as expected in the real-world environment. The goal is to ensure that the system meets all the predefined requirements, performs efficiently, and integrates smoothly with existing processes.

6.1 System Setup and Configuration

- **Infrastructure Setup**: Set up the development and production environments for the project. This includes installing and configuring CouchDB as the database for handling employee, leave, service request and desktop support data.
- **Database Indexing**: Optimize CouchDB with appropriate indexing to improve the performance of queries that fetch employee data, leave balances, leave records service request and desktop support reports.
- **Security Setup**: Role-based access control ensures that only authorized users (e.g., admins) can access specific functionalities.
- Integrating the frontend (Angular with HTML, CSS, and TypeScript) with the backend (CouchDB, Gemini Ai and EmailJs).

6.2 Code Development

- **Core Functionality Development**: Develop the core functionality of both **Employee Dashboard** and **Admin Dashboard** using Angular (HTML, CSS, TypeScript). This includes features for employees to view personal details, leave reports, desktop support and service request reports, and for admins to view aggregated leave summaries.

- **Error Handling and API Management**: Implement error handling mechanisms to ensure smooth user interactions and reliable Gemini API operations. Set up rate limiting to avoid performance issues during peak usage.

6.3 Testing and Validation

- **Unit and Integration Testing**: Conduct unit tests for individual components like the leave calculation logic and integration tests to validate the interaction between the employee dashboard, leave management, and CouchDB.

- **Cross-Browser and Regression Testing**: Ensure the application works across different browsers (Chrome, Firefox, etc.) and devices, while also performing regression tests to confirm that new changes do not break existing features..

6.4 User Training and Deployment

- **User Training**: Conduct training sessions and provide comprehensive documentation (manuals and video tutorials) for employees and admins to ensure they can effectively use the system. This includes training for leave requests, viewing reports, and managing service requests.
- **Staging Environment**: Before full deployment, implement a staging environment that mirrors production. This ensures a smooth transition by allowing final testing without affecting live data.

- **Go-Live Deployment**: Deploy the system into the live environment with minimal downtime, ensuring that employees can start using the dashboards and chatbot functionality immediately.

6.5 Data Migration and Integration

- **Data Migration**: If transitioning from a previous system, migrate historical employee data, leave records, and service requests into CouchDB. This ensures continuity in reporting and system operations.
- **Third-Party Integrations**: Integrate external services, such as Gemini AI for chatbot functionality, and ensure email notifications and other third-party systems work seamlessly with the dashboards.

7. CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

The **Rule-Based Decision Tree Chatbot with AI** project, integrating Angular, CouchDB, and Gemini AI, has been developed and deployed as per the outlined objectives. The system has undergone comprehensive unit and integration testing, ensuring its performance and reliability. Each module, including the **Employee Dashboard** and **Admin Dashboard**, has been thoroughly tested for functionality and accuracy. The AI-driven chatbot, designed to assist employees with leave applications and service requests, has been validated for its responsiveness and ease of use.

The system provides significant benefits for users by simplifying the process of applying for leave, submitting service requests, and viewing employee data. The decision tree logic effectively guides users through these tasks, ensuring accurate inputs and timely responses. Additionally, CouchDB ensures efficient data handling and storage, while the AI integration enhances the overall user experience. The **Admin Dashboard** offers detailed reports on employee activities and leave records, enabling easy monitoring and decision-making.

7.2 FEATURES

1. Simple, user-friendly interface with responsive design.
2. Pick lists and dropdowns to minimize user data entry.
3. Real-time validation and alerts for input errors.
4. Role-based security access for employees and administrators.
5. Optimized performance with NoSQL CouchDB for handling large datasets.
6. AI-driven decision tree for leave applications desktop support and service request handling.
7. Tracking and summarizing employee leave and service request statuses.
8. Dashboard view for admins to analyze data date-wise, month-wise, and more.
9. Seamless integration with Gemini AI for intelligent chatbot functionality.

7.3 FUTURE ENHANCEMENT

1. **Graphical Reporting**: Introduce visual reports and dashboards for admins to track trends in leave applications, service requests, and employee performance.
2. **Multilingual Support**: Extend the chatbot functionality to support multiple languages for a global employee base.
3. **Mobile App Integration**: Create a mobile version of the employee and admin dashboard for better accessibility.

7.4 LIMITATIONS

1. **Response Delays**: The system may experience delays in performance when there is high server load or network latency, especially if CouchDB or AI services are temporarily down.
2. **Concurrency Issues**: Handling multiple simultaneous service requests or leave applications might affect the real-time response of the chatbot or database transactions.

8. BIBLIOGRAPHY

8.1 Web References

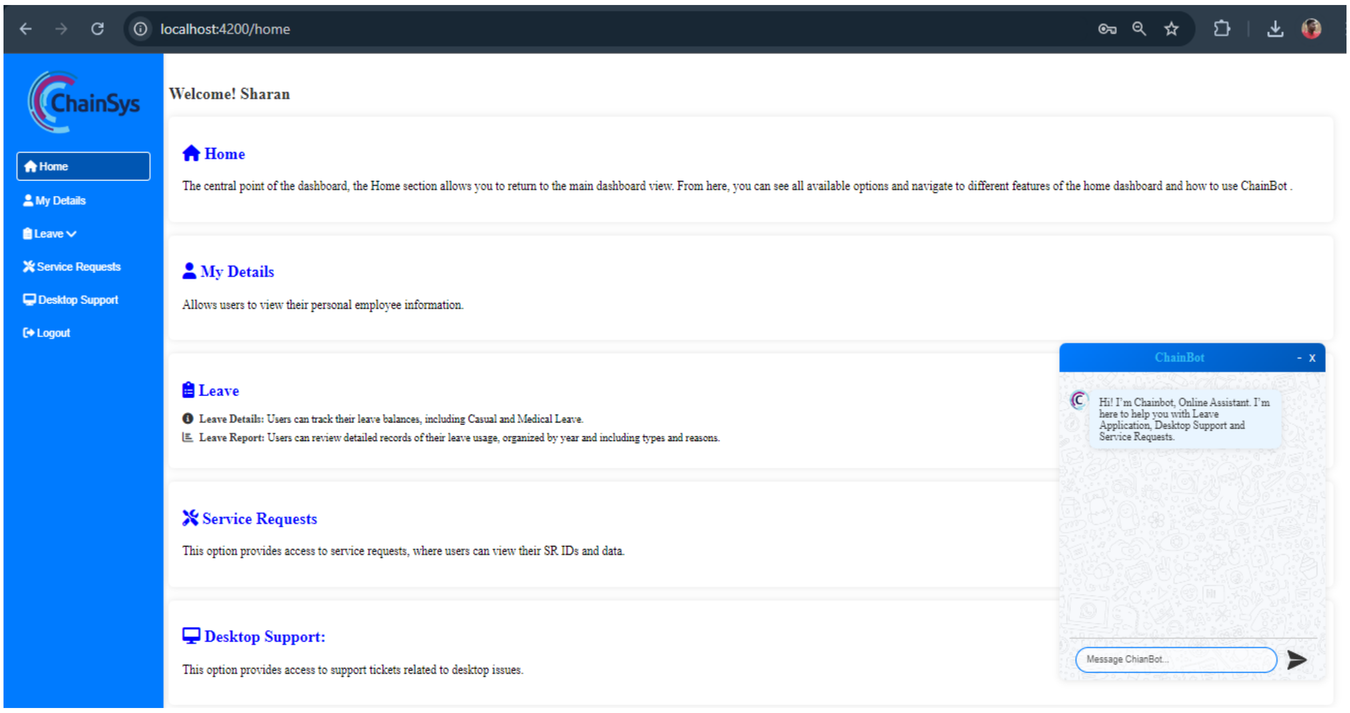
1. Angular : https://w3.p2hp.com/angular/angular_w3css.asp
2. HTML: https://www.w3schools.com/html/
3. CSS: https://www.w3schools.com/css/default.asp
4. TypeScript: https://www.w3schools.com/typescript/
5. CouchDB: https://docs.couchdb.org/en/stable/intro/index.html
6. Types of Chatbots: https://www.youtube.com/
7. Types of Chatbots: https://gettalkative.com/info/decision-tree-vs-ai-chatbots#:~:text=Rule%2Dbased%20bots%20were%20the,selection%20of%20keywords%20or%20phrases.
8. Pros and Cons of Chatbot: https://madewithweb.com/law-firm-essential-guide-to-machine-learning-chatbots/pros-and-cons-of-rule-based-vs-ai-chatbots/
9. Rule Based Chatbot: https://www.chatinsight.ai/chatbots/rule-based-chatbot/
10. How to Build Rule Based Chatbot: https://www.chatbees.ai/blog/rule-based-chatbots

9. APPENDICES

9.1 APPENDIX –A: PROJECT SCREEN SHOTS

9.1.1 Employee Dashboard:

9.1.1.1 Home Page:



Welcome! Sharan

Home

The central point of the dashboard, the Home section allows you to return to the main dashboard view. From here, you can see all available options and navigate to different features of the home dashboard and how to use ChainBot .

My Details

Allows users to view their personal employee information.

Leave

- **Leave Details:** Users can track their leave balances, including Casual and Medical Leave.
- **Leave Report:** Users can review detailed records of their leave usage, organized by year and including types and reasons.

Service Requests

This option provides access to service requests, where users can view their SR IDs and data.

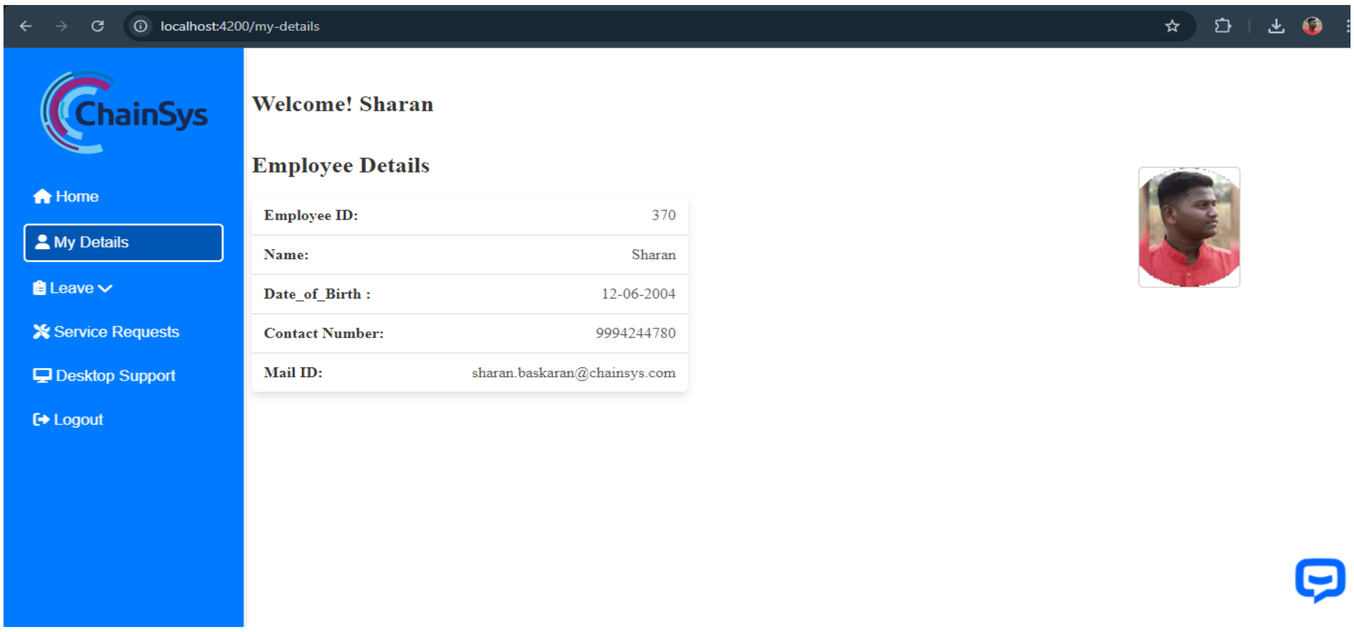
Desktop Support:

This option provides access to support tickets related to desktop issues.

Hi! I'm Chainbot, Online Assistant. I'm here to help you with Leave Application, Desktop Support and Service Requests.

Fig 9.1.1.1 Home Page

9.1.1.2 My Details Page:



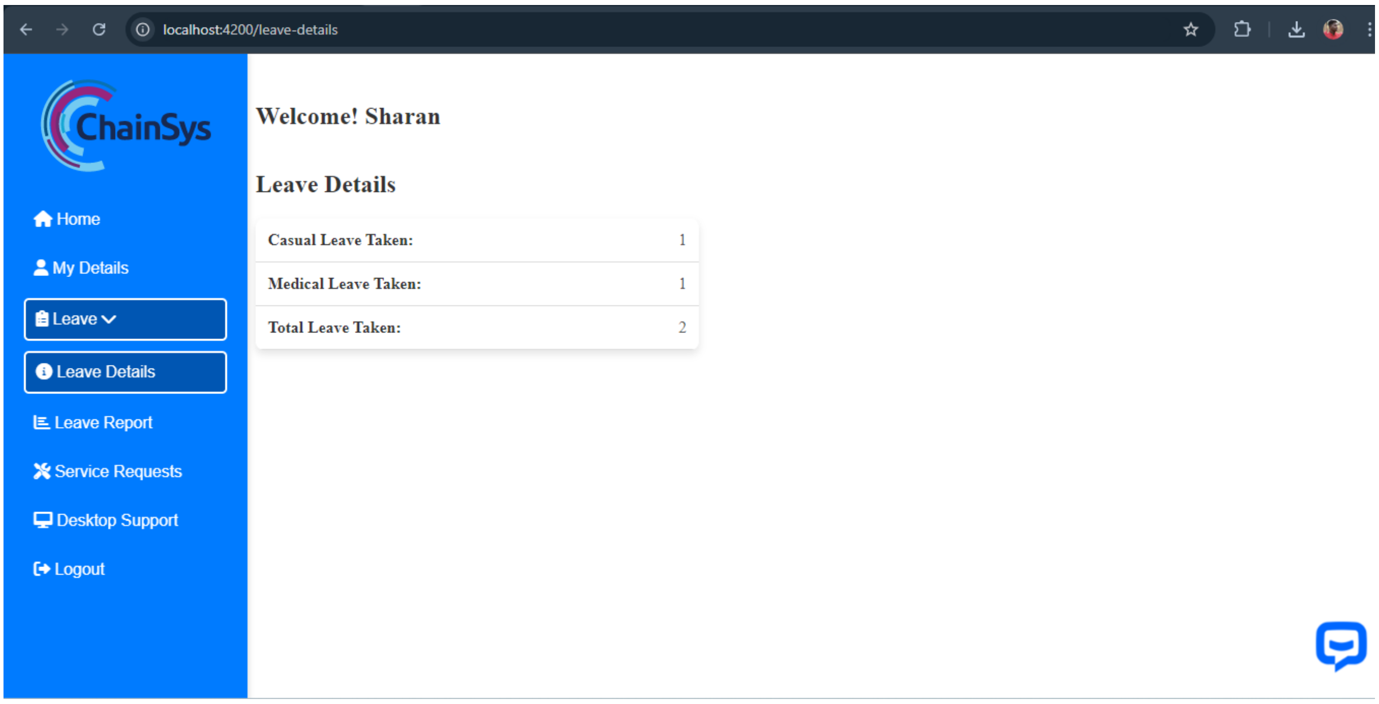
Welcome! Sharan

Employee Details

Employee ID:	370
Name:	Sharan
Date_of_Birth :	12-06-2004
Contact Number:	9994244780
Mail ID:	sharan.baskaran@chainsys.com

Fig 9.1.1.2 My Details Page

9.1.1.3 Leave Details Page:



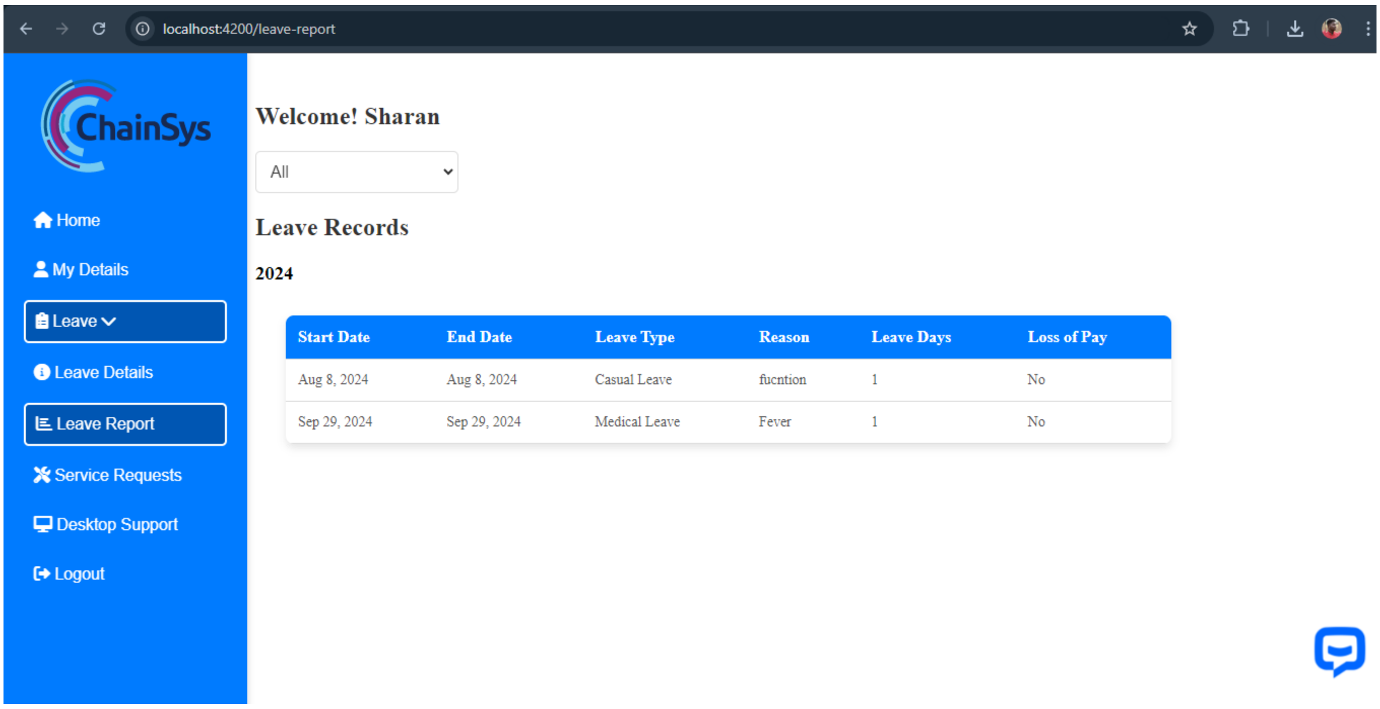
Welcome! Sharan

Leave Details

Casual Leave Taken:	1
Medical Leave Taken:	1
Total Leave Taken:	2

Fig 9.1.1.3 Leave Details Page

9.1.1.4 Leave Report Page:



Welcome! Sharan

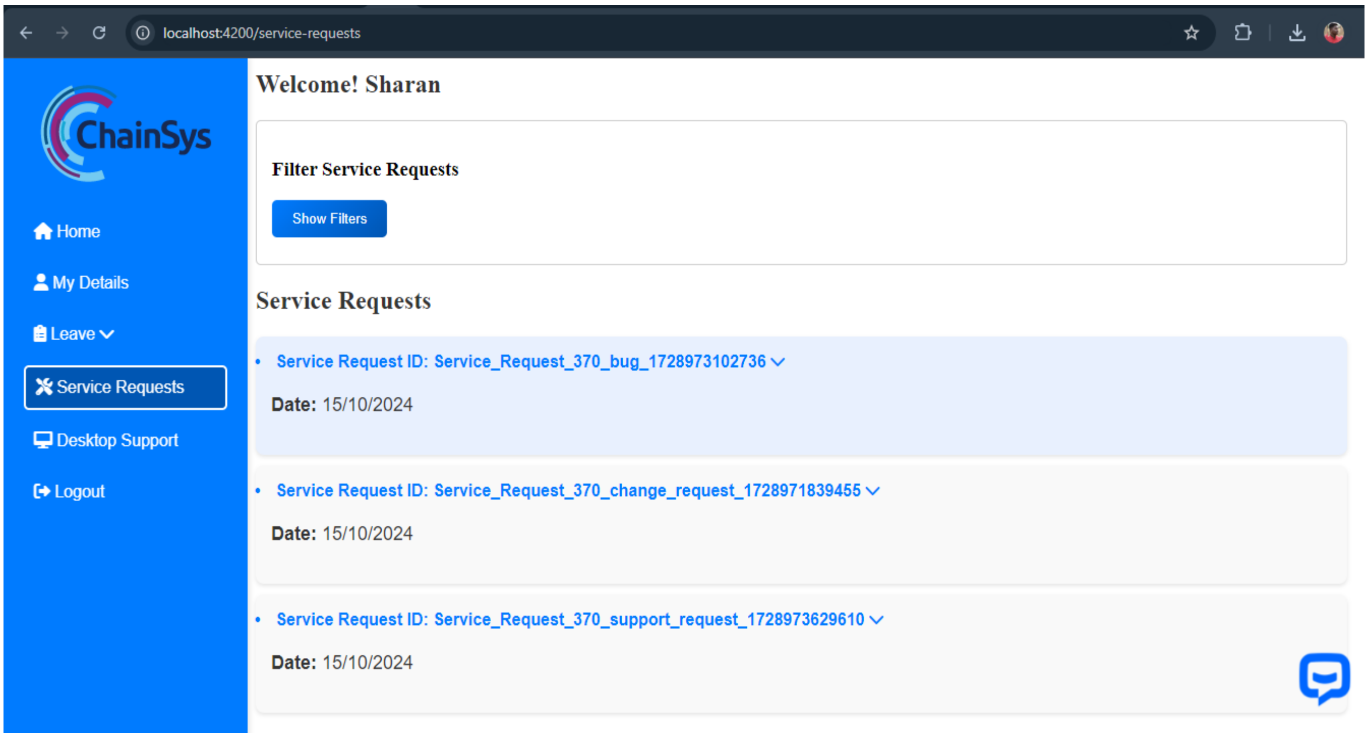
Leave Records

2024

Start Date	End Date	Leave Type	Reason	Leave Days	Loss of Pay
Aug 8, 2024	Aug 8, 2024	Casual Leave	fucntion	1	No
Sep 29, 2024	Sep 29, 2024	Medical Leave	Fever	1	No

Fig 9.1.1.4 Leave Report Page

9.1.1.5 Service Requests Page:



Welcome! Sharan

Filter Service Requests

Service Requests

- Service Request ID: Service_Request_370_bug_1728973102736

  Date: 15/10/2024

- Service Request ID: Service_Request_370_change_request_1728971839455

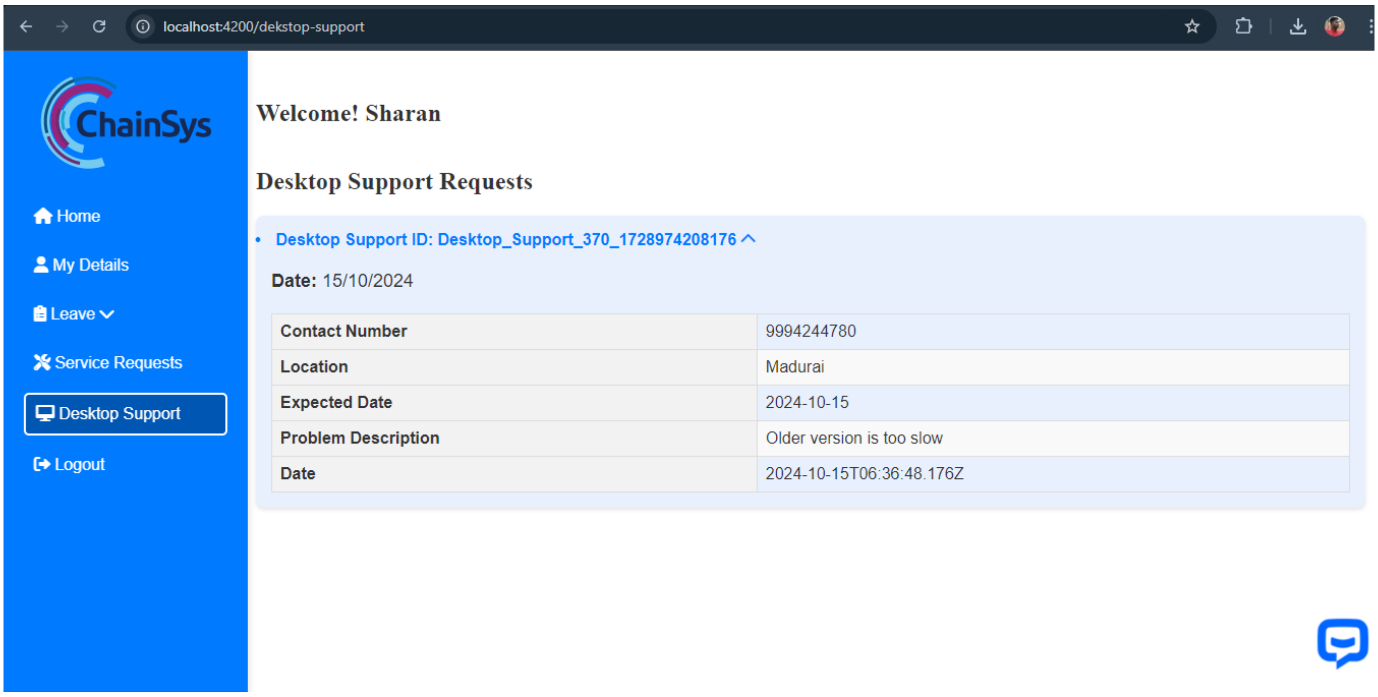
  Date: 15/10/2024

- Service Request ID: Service_Request_370_support_request_1728973629610

  Date: 15/10/2024

Fig 9.1.1.5 Service Request Page

9.1.1.6 Desktop Support Page:



Welcome! Sharan

Desktop Support Requests

- Desktop Support ID: Desktop_Support_370_1728974208176

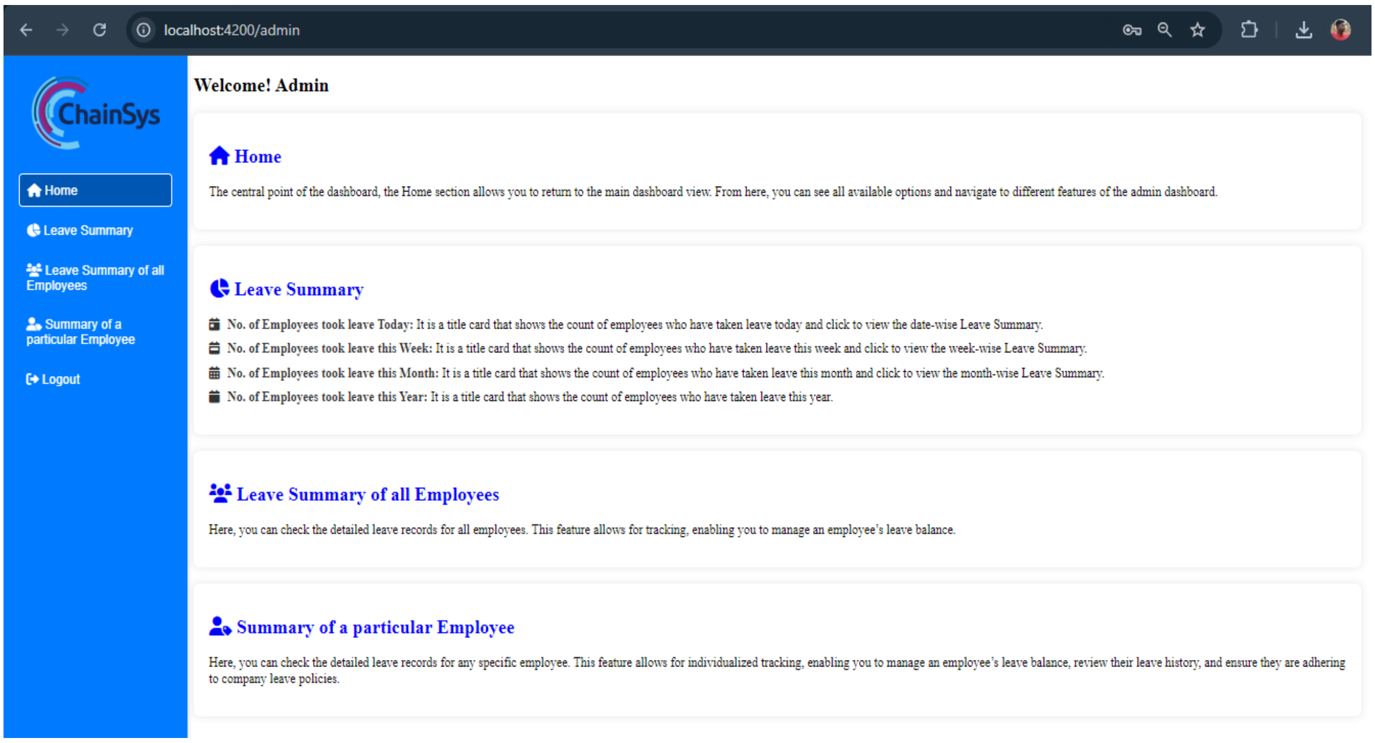
  Date: 15/10/2024

Contact Number	9994244780
Location	Madurai
Expected Date	2024-10-15
Problem Description	Older version is too slow
Date	2024-10-15T06:36:48.176Z

Fig 9.1.1.6 Desktop Support Page

9.1.2 Admin Dashboard:

9.1.2.1 Home Page:


localhost:4200/admin
ChainSys
Home
Leave Summary
Leave Summary of all Employees
Summary of a particular Employee
Logout

Welcome! Admin

Home

The central point of the dashboard, the Home section allows you to return to the main dashboard view. From here, you can see all available options and navigate to different features of the admin dashboard.

Leave Summary

- **No. of Employees took leave Today:** It is a title card that shows the count of employees who have taken leave today and click to view the date-wise Leave Summary.
- **No. of Employees took leave this Week:** It is a title card that shows the count of employees who have taken leave this week and click to view the week-wise Leave Summary.
- **No. of Employees took leave this Month:** It is a title card that shows the count of employees who have taken leave this month and click to view the month-wise Leave Summary.
- **No. of Employees took leave this Year:** It is a title card that shows the count of employees who have taken leave this year.

Leave Summary of all Employees

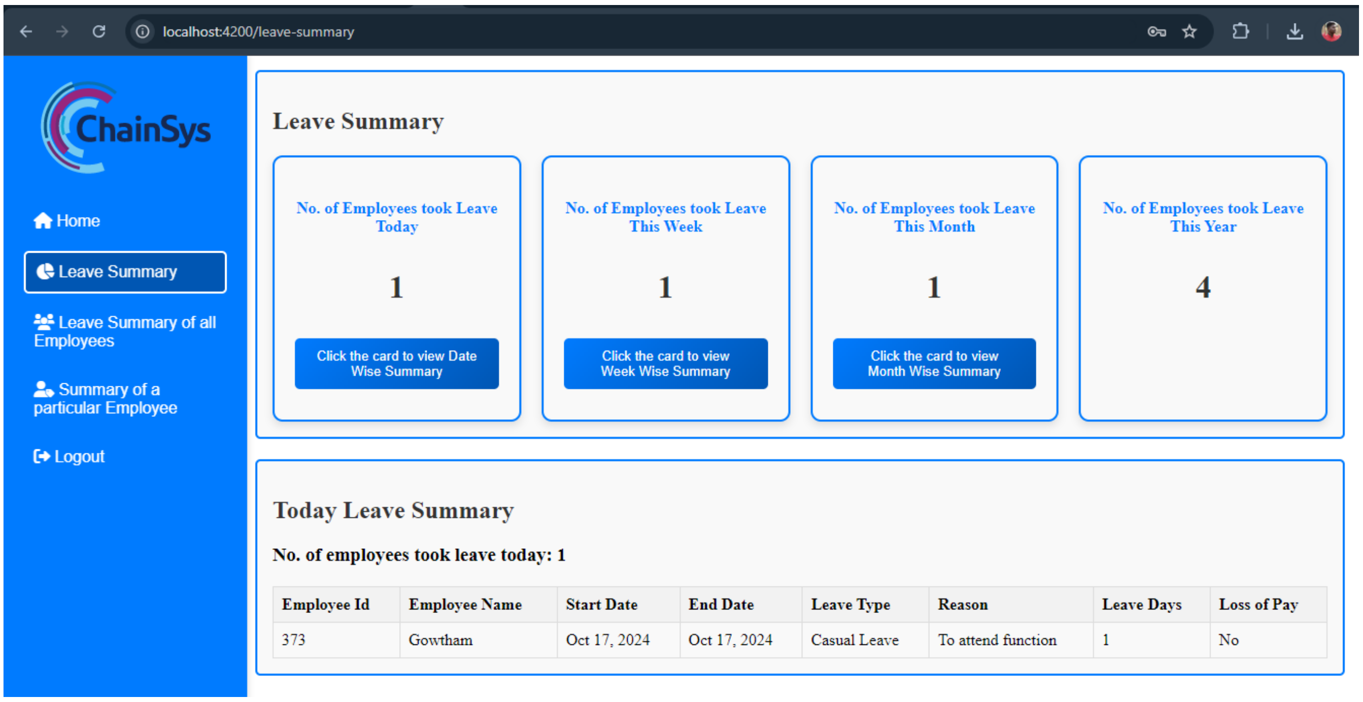
Here, you can check the detailed leave records for all employees. This feature allows for tracking, enabling you to manage an employee's leave balance.

Summary of a particular Employee

Here, you can check the detailed leave records for any specific employee. This feature allows for individualized tracking, enabling you to manage an employee's leave balance, review their leave history, and ensure they are adhering to company leave policies.

Fig 9.1.2.1 Home Page

9.1.2.2 Leave Summary:


localhost:4200/leave-summary
ChainSys
Home
Leave Summary
Leave Summary of all Employees
Summary of a particular Employee
Logout

Leave Summary

No. of Employees took Leave Today	No. of Employees took Leave This Week	No. of Employees took Leave This Month	No. of Employees took Leave This Year
1	1	1	4
Click the card to view Date Wise Summary	Click the card to view Week Wise Summary	Click the card to view Month Wise Summary	

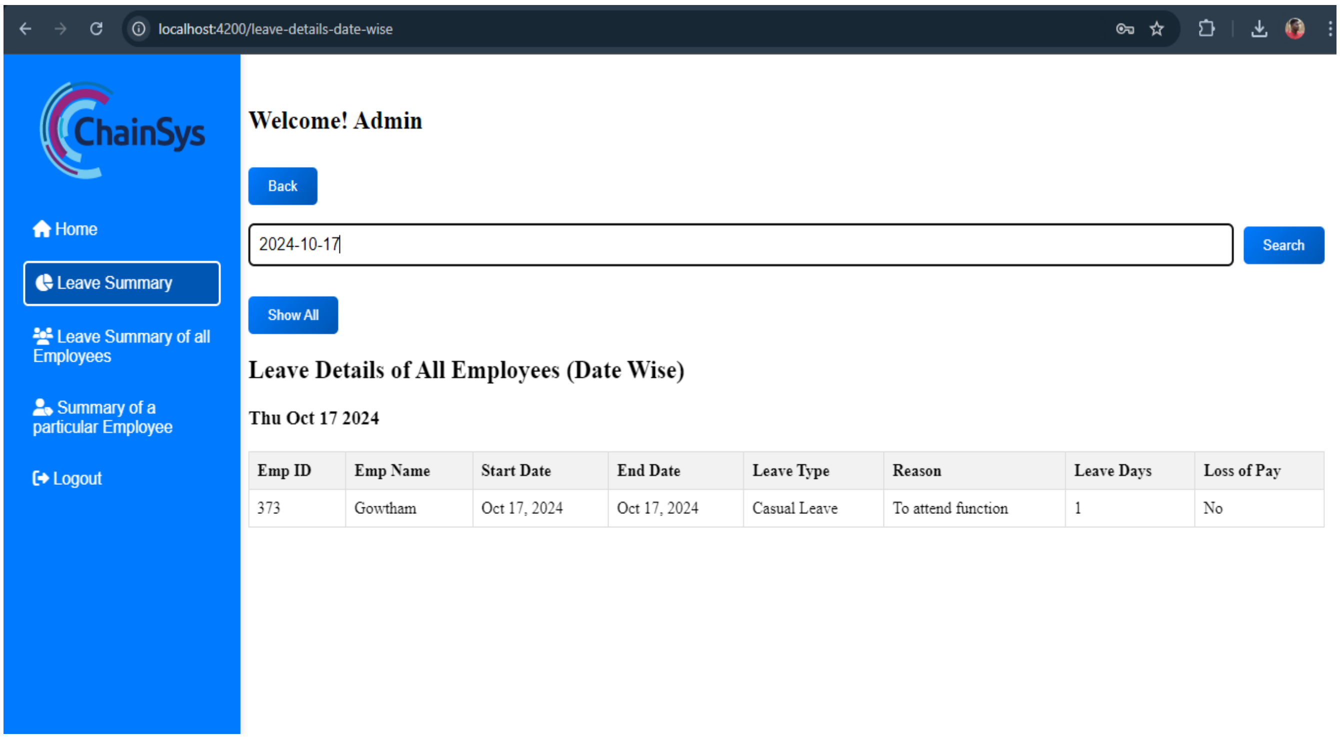
Today Leave Summary

No. of employees took leave today: 1

Employee Id	Employee Name	Start Date	End Date	Leave Type	Reason	Leave Days	Loss of Pay
373	Gowtham	Oct 17, 2024	Oct 17, 2024	Casual Leave	To attend function	1	No

Fig 9.1.2.2 Leave Summary Page

9.1.2.3 Leave Summary – Date Wise:


localhost:4200/leave-details-date-wise
ChainSys
Home
Leave Summary
Leave Summary of all Employees
Summary of a particular Employee
Logout
Back
Search
Show All

Welcome! Admin

2024-10-17

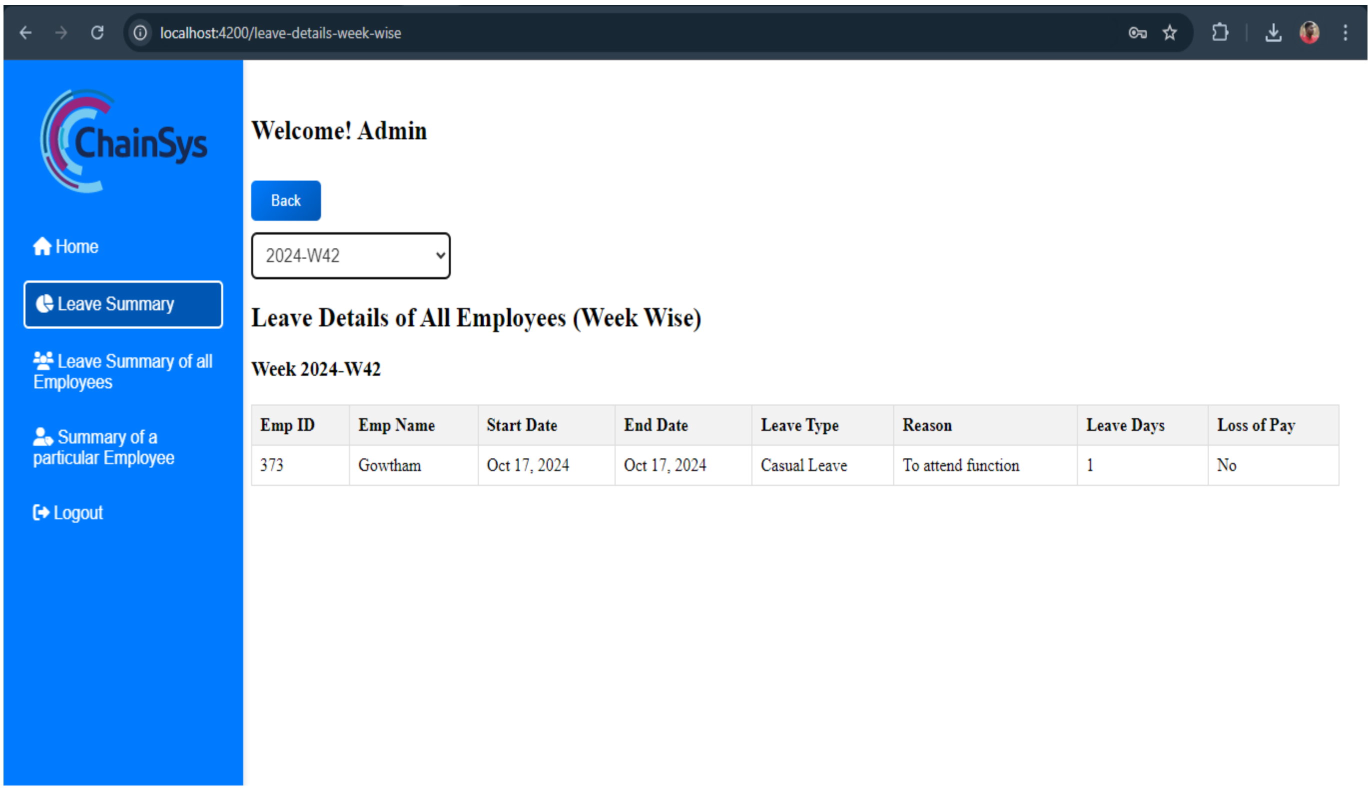
Leave Details of All Employees (Date Wise)

Thu Oct 17 2024

Emp ID	Emp Name	Start Date	End Date	Leave Type	Reason	Leave Days	Loss of Pay
373	Gowtham	Oct 17, 2024	Oct 17, 2024	Casual Leave	To attend function	1	No

Fig 9.1.2.3 Leave Summary – Date Wise Page

9.1.2.4 Leave Summary – Week Wise:


localhost:4200/leave-details-week-wise
ChainSys
Home
Leave Summary
Leave Summary of all Employees
Summary of a particular Employee
Logout
Back

Welcome! Admin

2024-W42

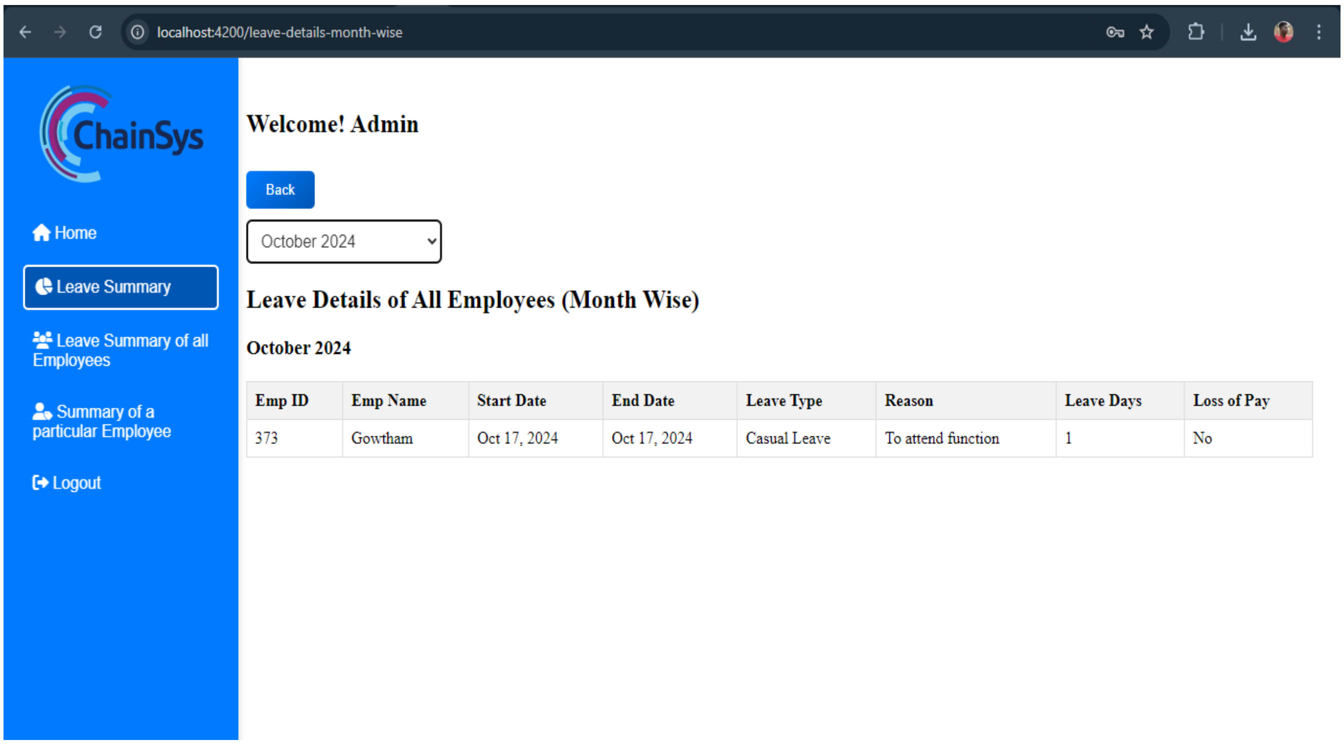
Leave Details of All Employees (Week Wise)

Week 2024-W42

Emp ID	Emp Name	Start Date	End Date	Leave Type	Reason	Leave Days	Loss of Pay
373	Gowtham	Oct 17, 2024	Oct 17, 2024	Casual Leave	To attend function	1	No

Fig 9.1.2.4 Leave Summary – Week Wise Page

9.1.2.5 Leave Summary – Month Wise:


localhost:4200/leave-details-month-wise
ChainSys
Home
Leave Summary
Leave Summary of all Employees
Summary of a particular Employee
Logout
Back
October 2024

Welcome! Admin

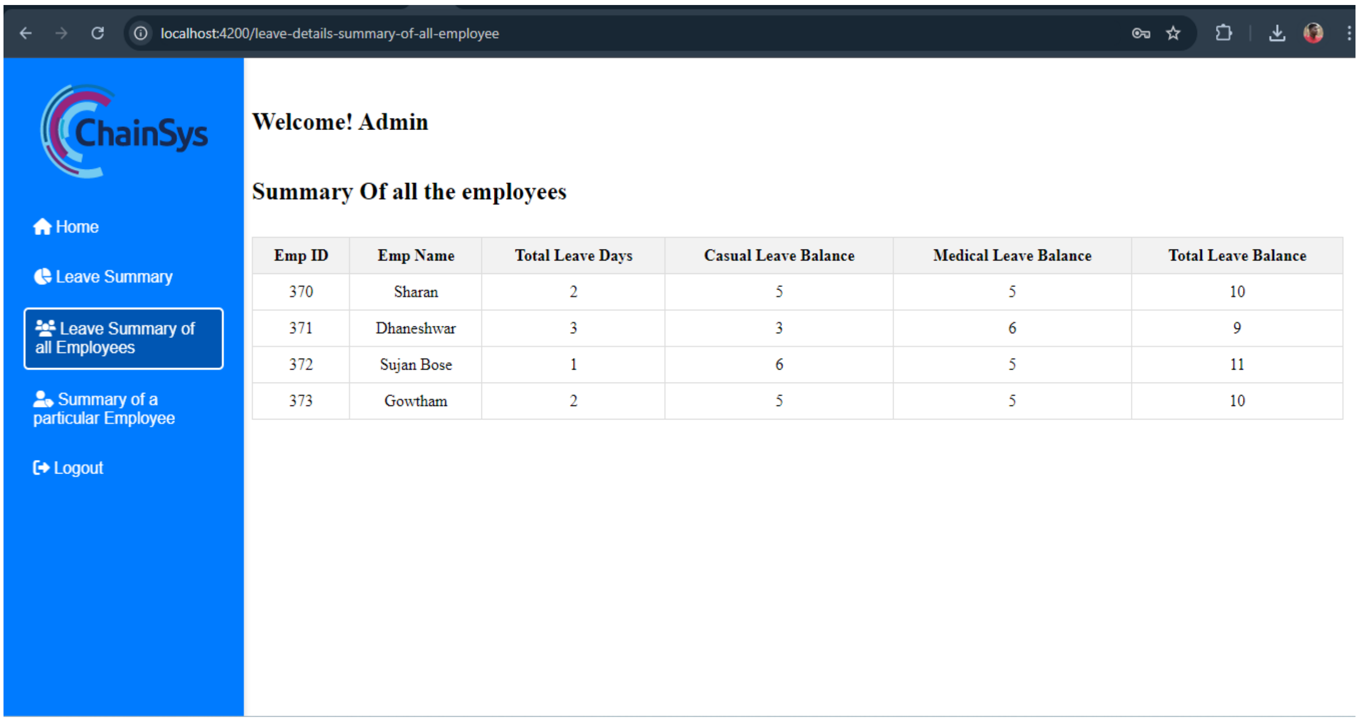
Leave Details of All Employees (Month Wise)

October 2024

Emp ID	Emp Name	Start Date	End Date	Leave Type	Reason	Leave Days	Loss of Pay
373	Gowtham	Oct 17, 2024	Oct 17, 2024	Casual Leave	To attend function	1	No

Fig 9.1.2.5 Leave Summary – Month Wise Page

9.1.2.6 Leave Summary of all Employees


localhost:4200/leave-details-summary-of-all-employee
ChainSys
Home
Leave Summary
Leave Summary of all Employees
Summary of a particular Employee
Logout

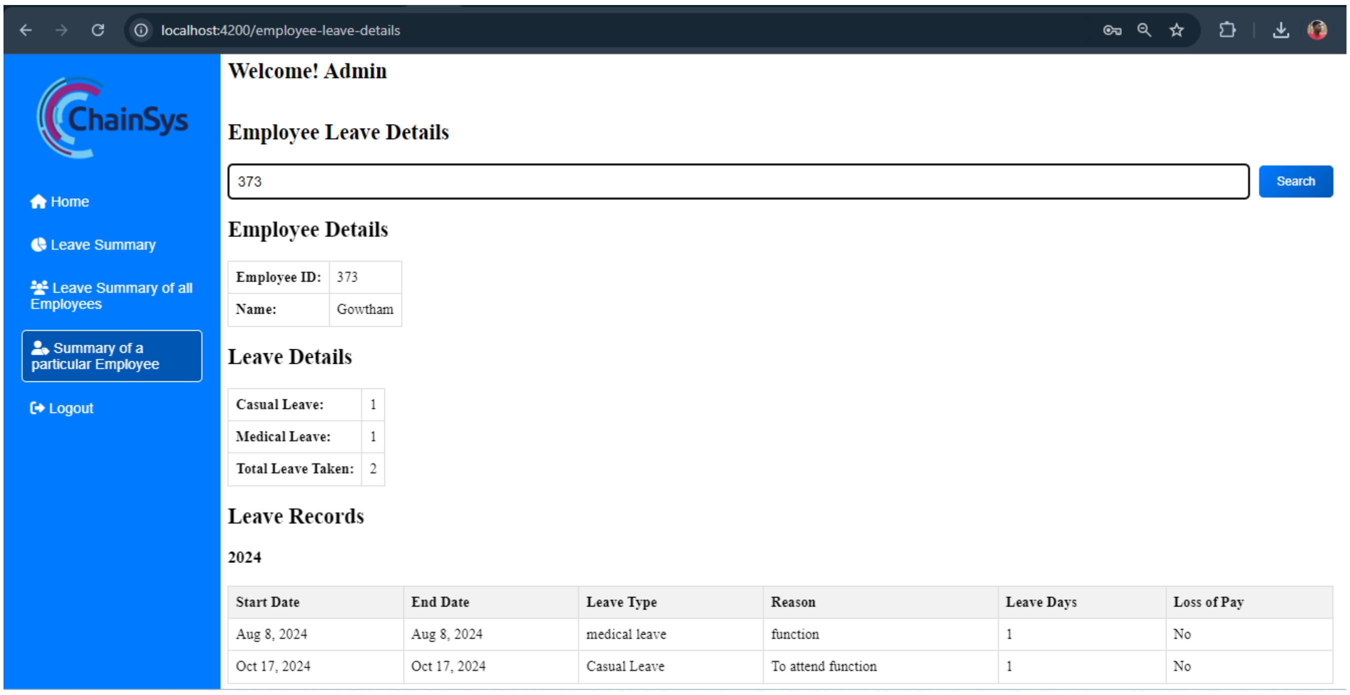
Welcome! Admin

Summary Of all the employees

Emp ID	Emp Name	Total Leave Days	Casual Leave Balance	Medical Leave Balance	Total Leave Balance
370	Sharan	2	5	5	10
371	Dhaneshwar	3	3	6	9
372	Sujan Bose	1	6	5	11
373	Gowtham	2	5	5	10

Fig 9.1.2.6 Leave Summary of all the employees Page

9.1.2.7 Summary of a particular Employee:


localhost:4200/employee-leave-details
ChainSys
Home
Leave Summary
Leave Summary of all Employees
Summary of a particular Employee
Logout
Search

Welcome! Admin

Employee Leave Details

373

Employee Details

Employee ID:	373
Name:	Gowtham

Leave Details

Casual Leave:	1
Medical Leave:	1
Total Leave Taken:	2

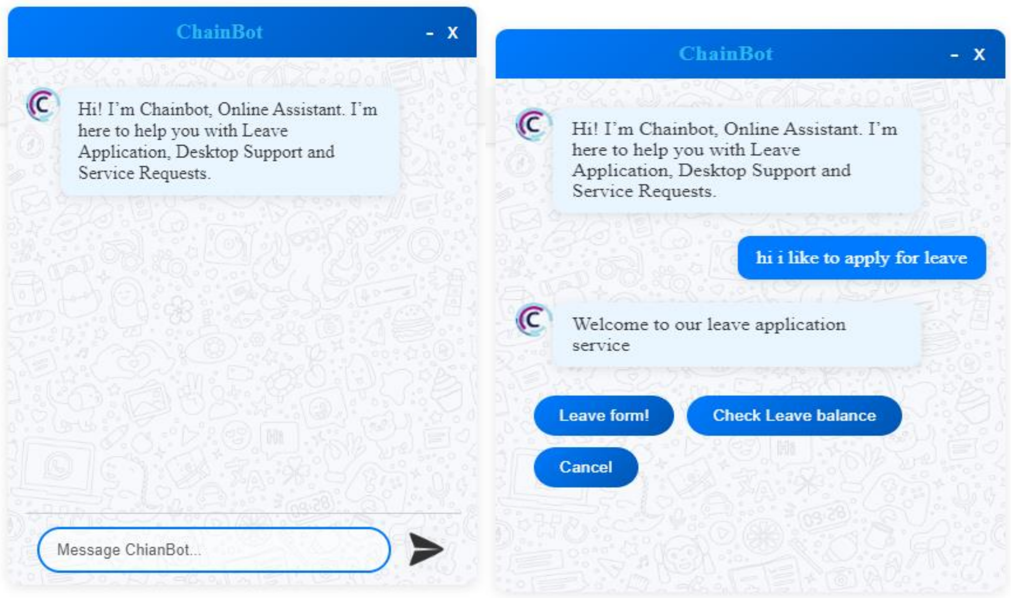
Leave Records

2024

Start Date	End Date	Leave Type	Reason	Leave Days	Loss of Pay
Aug 8, 2024	Aug 8, 2024	medical leave	function	1	No
Oct 17, 2024	Oct 17, 2024	Casual Leave	To attend function	1	No

Fig 9.1.2.7 Leave Summary of a particular employee Page

9.1.3 ChainBot:



Hi! I'm Chainbot, Online Assistant. I'm here to help you with Leave Application, Desktop Support and Service Requests.

Hi! I'm Chainbot, Online Assistant. I'm here to help you with Leave Application, Desktop Support and Service Requests.

hi i like to apply for leave

Welcome to our leave application service

Fig 9.1.3 ChainBot UI

9.1.4 Login Page:

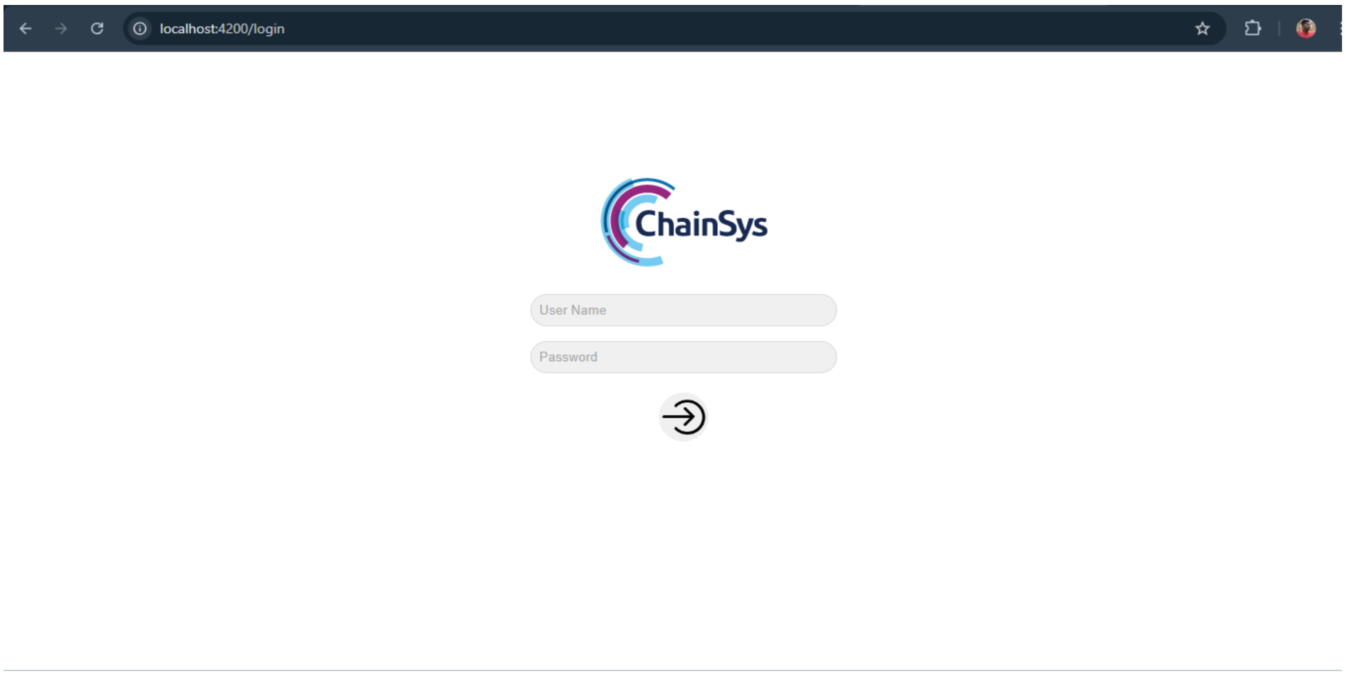
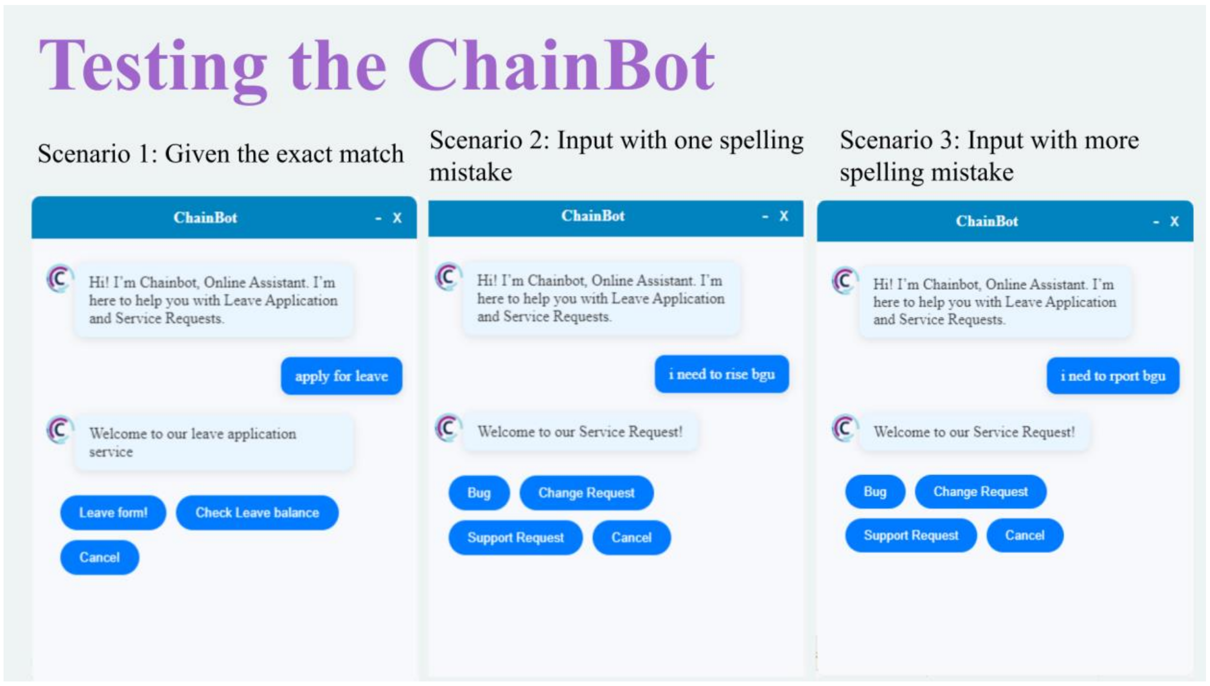


Fig 9.1.4 Login Page

9.2 APPENDIX –A: UNIT TESTING SCREENSHOTS:

9.2.1 Testing the ChainBot:



Testing the ChainBot

Scenario 1: Given the exact match

ChainBot - X

Hi! I'm Chainbot, Online Assistant. I'm here to help you with Leave Application and Service Requests.

apply for leave

Welcome to our leave application service

Scenario 2: Input with one spelling mistake

ChainBot - X

Hi! I'm Chainbot, Online Assistant. I'm here to help you with Leave Application and Service Requests.

i need to rise bgu

Welcome to our Service Request!

Scenario 3: Input with more spelling mistake

ChainBot - X

Hi! I'm Chainbot, Online Assistant. I'm here to help you with Leave Application and Service Requests.

i ned to rport bgu

Welcome to our Service Request!

Fig 9.2.1 Testing the ChainBot